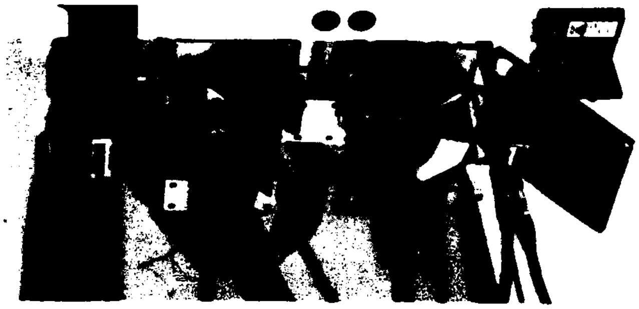
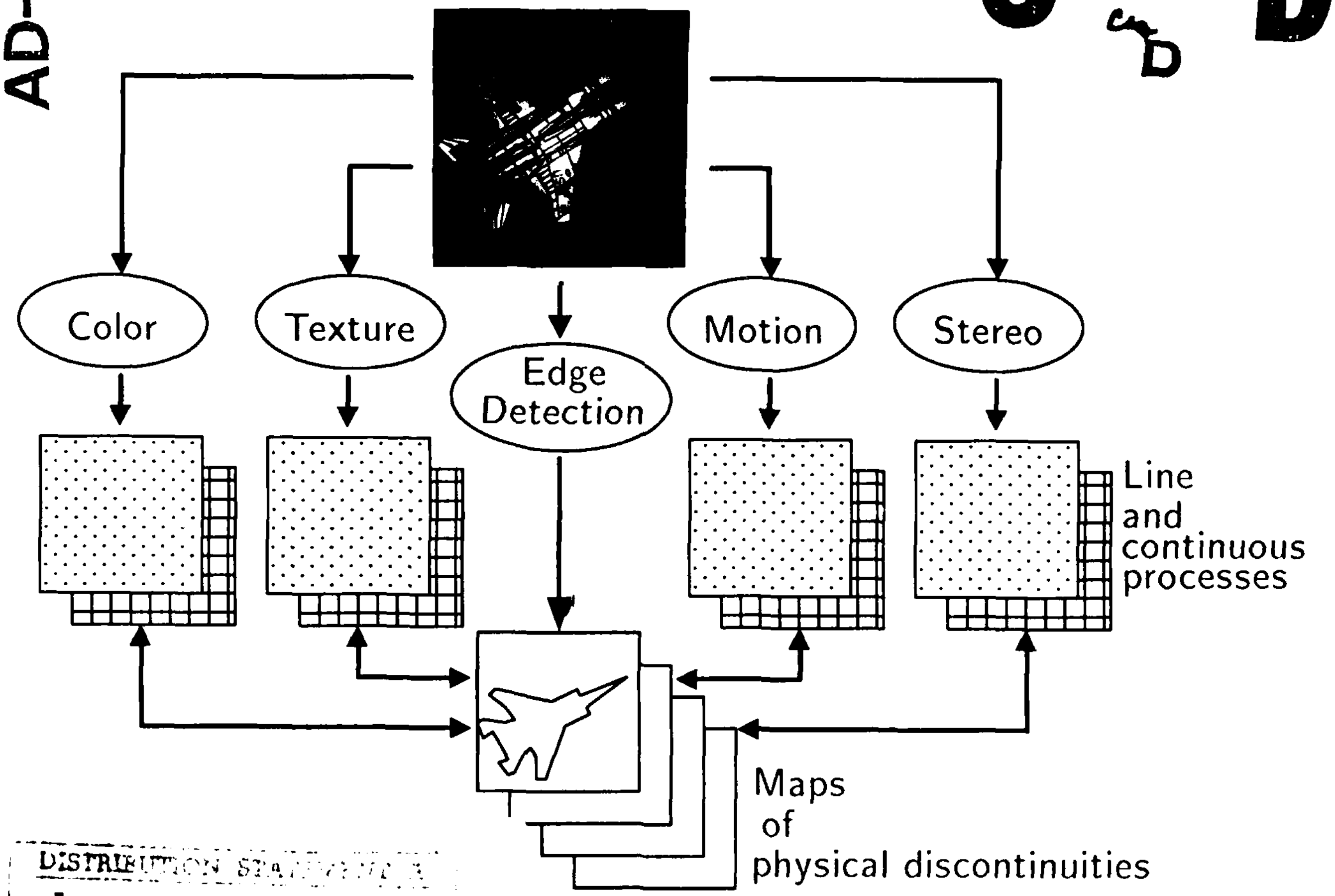
Volume II

PROCEEDINGS:

Image Understanding Workshop

(2)




Color
Texture
Edge Detection
Motion
Stereo
Line and continuous processes
Maps of physical discontinuities

Sponsored by:

Defense Advanced Research Projects Agency
Information Science and Technology Office

DARPA

SC

April 1988

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Proceedings: Image Understanding Workshop April 1988		5. TYPE OF REPORT & PERIOD COVERED ANNUAL TECHNICAL February 1987-April 1988
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Lee S. Baumann (Editor)		8. CONTRACT OR GRANT NUMBER(s) N00014-86-C0700
9. PERFORMING ORGANIZATION NAME AND ADDRESS SCIENCE APPLICATIONS INTERNATIONAL CORPORATION 1710 Goodridge Drive McLean, VA 22102		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS ARPA Order 5605
11. CONTROLLING OFFICE NAME AND ADDRESS Defense Advanced Research Projects Agency 1400 Wilson Blvd Arlington, VA 22209		12. REPORT DATE April 1988
		13. NUMBER OF PAGES 1165 (2 Vols.)
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

APPROVED FOR PUBLIC RELASE, DISTRIBUTION UNLIMITED

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Digital Image Processing; Image Understanding; Scene Analysis;
Edge Detection; Image Segmentation; CCDArrays; CCD Processors

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This document contains the annual progress reports and technical papers presented by the research activities in the Image Understanding, sponsored by the Information Science & Technology Office, Defense Advanced Research Projects Agency. The papers were presented at a workshop conducted on 6-8 April 1988, in Cambridge, Massachusetts. Also included are copies of invited papers presented at the workshop and additional technical papers from the research activities which were not presented due to lack of time but are germane to this research field.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

Image Understanding Workshop

Proceedings of a Workshop
Held at
Cambridge, Massachusetts

April 6-8, 1988

Volume II

Sponsored by:

Defense Advanced Research Projects Agency
Information Science and Technology Office

This document contains copies of reports prepared for the DAR ?A Image Understanding Workshop. Included are results from both the basic and strategic computing programs within DARPA/ISTO sponsored projects.

APPROVED FOR PUBLIC RELEASE
DISTRIBUTION UNLIMITED

The views and conclusions contained in this document are those of the authors and should not be interpreted as necesarily representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the United States Government.

Distributed by

Morgan Kaufmann Publishers, Inc.

2929 Campus Drive

San Mateo, California 94403

ISBN 0-934613-68-0

Printed in the United States of America

TABLE OF CONTENTS

VOLUME I

SECTION I - PROGRAM REVIEWS BY PRINCIPAL INVESTIGATORS

TABLE OF CONTENTS

TABLE OF CONTENTS

TABLE OF CONTENTS

TABLE OF CONTENTS

Page

SECTION II - TECHNICAL REPORTS PRESENTED (Continued)

VOLUME II

SECTION III - OTHER TECHNICAL REPORTS

TABLE OF CONTENTS

Page

SECTION III - OTHER TECHNICAL REPORTS (Continued)

TABLE OF CONTENTS

TABLE OF CONTENTS

Page

SECTION III - OTHER TECHNICAL REPORTS (Continued)

TABLE OF CONTENTS

Page

SECTION III - OTHER TECHNICAL REPORTS (Continued)

TABLE OF CONTENTS

Page

SECTION III - OTHER TECHNICAL REPORTS (Continued)

TABLE OF CONTENTS

AUTHOR INDEX

AUTHOR INDEX (Continued)

AUTHOR INDEX (Continued)

AUTHOR INDEX (Continued)

Acknowledgments

This workshop was organized at the direction of Ltc. Robert L. Simpson, Program Manager for Machine Intelligence in the Information Science and Technology Office of the Defense Advanced Research Projects Agency (DARPA). The main theme of the 1988 workshop, the 18th in this DARPA sponsored series of meetings on Image Understanding and Computer Vision, is "Parallel Architectures and Algorithms." Sessions are planned to cover new vision techniques in prototype vision systems for manufacturing, navigation, cartography, and photointerpretation. As usual the main objectives of the workshop are to review the latest research results in the DARPA IU research progam, help keep the government research community aware of evolving technology and most importantly to exchange ideas, needs and trends in computer vision research. Dr. Tomaso Poggio of the Artificial Intelligence Laboratory at the Massachusetts Institute of Technology coordinated the technical program for this workshop.

As part of the agenda principal investigator's reports and technical papers are scheduled to be presented from MIT, University of Southern California, Stanford University, University of Maryland, Carnegie-Mellon University, SRI International, University of Massachusetts, Rochester University, Columbia University, Hughes Research Laboratory, General Electric Research and Development Center, Advanced Decision Systems, and Honeywell Systems and Research Center. In addition, invited papers are planned for presentation from the University of Illinois, NYU - Courant Institute, University of Washington, University of Pennsylvania, David Sarnoff Research Center, and Schlumber Palo Alto Research Center.

As is the normal practice for these workshops all technical reports submitted by participating organizations are published in this comprehensive proceedings available for distribution to the attendees, including both those presented at the workshop and those for which lack of time precluded presentation.

The figure appearing on the cover was prepared by Dr. J. Little of the MIT AI Laboratory and represents a block diagram of the integration stage of the MIT Vision Machine. For details see "The Vision Machine", authored by Dr. T. Poggio, et. al., in these proceedings. The cover layout was prepared by Mr. Tom Dickerson of the Science Applications International Corporation (SAIC) graphic arts staff. Appreciation is also extended to Ms. Bethany Moss of SAIC for her work in compiling the papers into this proceedings and handling the mailings associated with the 1988 I. U. Workshop.

SECTION III

OTHER TECHNICAL REPORTS

MULTIRESOLUTION AERIAL IMAGE INTERPRETATION

Teresa M. Silberberg

Hughes Artificial Intelligence Center
23901 Calabasas Road
Calabasas, CA 91302

Abstract

The interpretation system presented in this paper uses multiple image resolutions to detect objects in large, cluttered, high resolution, aerial images. Lower resolutions are processed first so attention is focused in the higher resolutions to those areas where there is evidence of the objects. The system provides a framework for modeling objects at each resolution using a variety of complementary image features. The approach is both data-driven and model-driven, utilizes hypothesis generation and verification, and employs evidential reasoning to evaluate the hypotheses. The system has been exercised on two aerial images; one consisting of a single submarine and analyzed using three resolutions, and the other consisting of three airplanes and analyzed using two resolutions.

1. INTRODUCTION

The interpretation of a large, cluttered, high resolution image can be facilitated by examining features extracted from multiple resolutions of the image. The objects of interest are modeled at each resolution in terms of features that can be used to provide evidence for the objects. By examining lower resolutions during the initial stages of image interpretation, object hypotheses can be made based on large, prominent features without the complications of clutter and object detail present at higher resolutions. Given these initial hypotheses, higher resolutions are examined only in those areas in which objects of interest are expected. By focusing in on the objects, a more efficient search of the cluttered, high resolution images is achieved.

In the system described in this paper, an object is modeled according to its appearance in the image using two kinds of features: salient features that create initial object hypotheses, and supporting features that provide evidence for the hypothesized object. At each resolution, hypotheses can result from two types of processing. First, a hypothesis may result from a feature that is due to an instance of the object. Second, a hypothesis may result from objects that exist at a lower resolution (e.g., an object projects itself down to a higher resolution) or at the same resolution (e.g., an object hypothesizes another within the same resolution). In either case, the object creates hypotheses in accordance with the model that specifies the confirming evidence.

The process by which areas in multiresolution images are interpreted as scene objects can be broken into several modules. Interactions between the resolutions during interpretation, communication and feedback between selected modules, and infusion of scene knowledge into appropriate modules have the effect of improving the robustness of the interpretation process. In a simple example using a single resolution there are two modules: the Symbolic Description Module and the Interpretation Module. The Symbolic Description Module transforms the original image into symbolic descriptions. Any of the algorithms can use ancillary information to refine parameter values. For example, the image resolution and range to an object will affect the size of the object in the image; this information can be used to guide a procedure that looks for lines that are the length of the object. The Interpretation Module applies ancillary information and the object models to the symbolic descriptions to provide a consistent description of the scene. As a resolution is analyzed, interpretations from previously processed resolutions are made available, and queries of other resolutions can be made. The two modules interact until the best interpretation is found for that resolution. (This description is similar to a context dependent target recognition system described in Silberberg [10].)

Among the issues to be addressed in the development of a multiresolution image interpretation system are the effective transformation of image data to symbolic descriptions, the modeling of the objects and their parts, the representation of symbolic and ancillary information, and the reasoning used to apply the knowledge at the appropriate resolutions and to guide hypothesis interaction within and between resolutions. The image interpretation system does not follow an algorithmic approach; that is, it does not consist of a specific ordering of procedures. Instead, the system dynamically chooses procedures based on its current state. The approach is both data-driven and model-driven, utilizes hypothesis generation and verification, and employs evidential reasoning to evaluate the hypotheses. The system has been exercised on two aerial images; one consisting of a single submarine and analyzed using three resolutions, and the other consisting of three airplanes and analyzed using two resolutions.

Several single resolution image interpretation systems have been developed. See, for example, McKeown et al. [6], Brooks [2], Hanson and Riseman [4], and the survey by Binford [1]. A system for object recognition using model and image pyramids is described by Neveu et al. [7]. In the distributed system of Tan and Martin [11], a multiresolution pyramid is used to detect and track moving objects.

In Section 2, we describe the data structure used to represent the initial symbolic descriptions and the resulting hypothesized scene objects. Section 3 presents a description of the multiresolution image interpretation system. The performance of the system on two images follows in Section 4. Conclusions and future work are discussed in Section 5.

2. MULTIRESOLUTION SYMBOLIC PIXEL ARRAY

An effective structure for representing image-based data must support operations that retrieve: (1) pixel properties, such as intensity and edge magnitude, and the object hypotheses that exist at a pixel; (2) object properties, such as area and compactness for regions, or length and orientation for lines; and (3) object spatial relationships, such as adjacency and nearness. Furthermore, retrieval of resolution-specific data must be supported. We use a data structure called the Multiresolution Symbolic Pixel Array (MSPA) which organizes the images and features by resolution and allows efficient retrieval of pixel and object properties and object spatial relationships within and between resolutions.

The MSPA is an extension of the single resolution Symbolic Pixel Array (Payton [8]). Each pixel property is stored as a full-sized array, and object extent in the image is stored as a binary array with x and y offsets. The binary array and its offsets are referred to as a *virtual array*. In the MSPA, pixel properties are retrieved by indexing into the appropriate full-sized array at a specified resolution. Object properties are computed as they are needed and then stored explicitly with the hypothesized object. A list of objects that exist at a given location in a specified level is determined by referencing each virtual array at that location. Objects that have been hypothesized in the same location but at a different resolution can be found by transforming the location coordinates appropriately.

By performing logical operations on the virtual arrays, we can compute spatial relationships quickly. Such spatial relationships include: find-distance-from-object-to-point, compute-distance-between-objects, find-objects-on-object, and is-object-near-point. Since the system is object-oriented, operations take the form of messages passed to the MSPA. Appropriate arguments to these spatial operations include one or more objects, x and y coordinates, allowed object types, and a list of one or more resolutions. As an example, the arguments for find-objects-on-object are: the base object, a list of resolutions to be examined, specific objects to ignore, allowed object types, and disallowed object types. If the list of resolutions is empty, then all resolutions are checked. If the base object does not exist at a specified resolution, then an object representing the base object is created temporarily. As a second example, the arguments for compute-distance-between-objects are: object$_1$, object$_2$, and the resolution, R, in which the distance is interpreted. The distance is computed at the nearest resolution to R in which at least one of the objects exist. If the other object does not exist at that resolution, it is created temporarily. The default for R is the highest resolution, and the distance must be returned in terms of R.

3. MULTIRESOLUTION INTERPRETATION

This section provides a description of the multiresolution image interpretation system. First, object interpretation using evidence gathering and evaluation is summarized. A more complete description can be found in Silberberg [10]. Next, the organization of the objects in an interpretation structure and the use of this structure to guide the interpretation of a single image is presented. Finally, multiresolution interpretation and hypothesis interaction are described.

Evidence gathering and evaluation

The interpretation approach is object-oriented, that is, each symbolic description is initially hypothesized or *instantiated* as one of the scene objects. The instantiated object then gathers information which provides evidence for or against the hypothesis. The scene knowledge, represented as object models and associated rules, directs the gathering of the evidence. If enough evidence is found, the hypothesis then becomes *established*, and the degree to which all of the evidence confirms or disconfirms the hypothesis is computed.

Each object model is characterized by attributes and desirable relationships between these attributes. The presence or absence of each attribute provides evidence that confirms or disconfirms an instantiation of that object. Attributes and their relationships are represented in the model using *slots* and *rules* that specify restrictions and relationships between these slots. The properties of each slot specify the type and number of the objects that can fill the slot as well as methods for filling the slot. A slot can be filled by finding an existing object or by hypothesizing a new object. The rules represent a collection of information that determines if a hypothesized symbolic description should be instantiated as a particular scene object and how to combine the evidence to compute a confidence. The evaluation and combination of uncertain evidence follows the method set forth in Mycin [3], that is, we maintain an overall confidence measure computed from a measure of belief and a measure of disbelief.

Interpretation of a single resolution

During interpretation, it is desirable to withhold commitment to an interpretation if there is insufficient evidence so that complex analysis of an incorrect interpretation is avoided. Additionally, rather than attempting interpretation of a symbolic description directly as an expected object, it is often more appropriate to gather pieces of evidence which can then hypothesize that object. A general-to-specific hierarchy can be utilized in realizing the former objective; a network that specifies support can be helpful in realizing the latter.

The structure used in representing the scene knowledge is a semantic network which is a directed graph consisting of a set of vertices and a set of labeled edges between pairs of vertices. In this representation, a vertex is an object type, and the label on an edge represents either an "a-kind-of" or an "in-support-of" relationship. Figure 1 shows the representation of objects in the submarine example. The solid edges represent the "a-kind-of" relationship ("shadow is a-kind-of region-scene-object"). Broken edges represent possible object dependencies where the object at the tail of the edge provides support for the object at the head ("shadow is in-support-of submarine").

Initially, each of the original symbolic descriptions is hypothesized as the most general object class (scene-object in Figure 1). For each hypothesis, object slots are filled by known objects or hypothesized new objects. In the event that one or more objects hypothesize another object, the in-support-of relationships are utilized. Objects that are in support of another can singly or jointly hypothesize that other object.

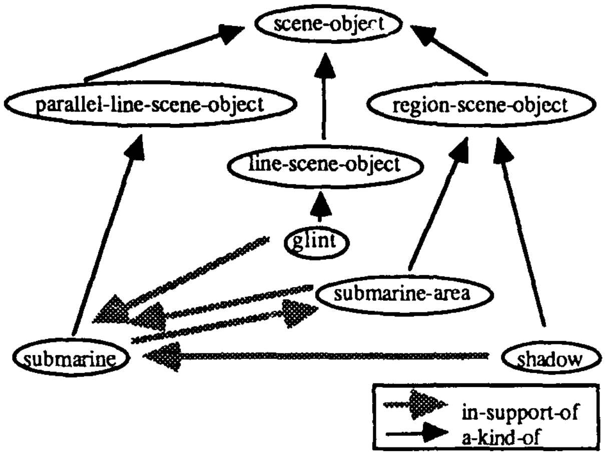


Figure 1. The semantic network representation for a submarine image. (The network is similar to that used in the experiments.)

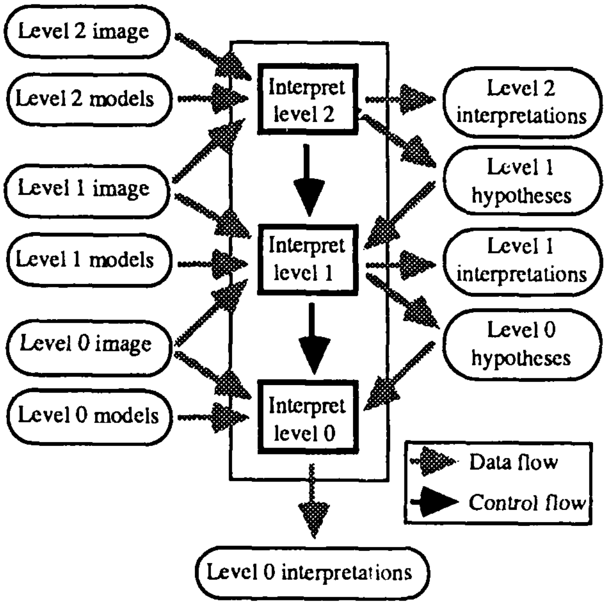


Figure 2. The interpretation process for 3 levels. Interpretation begins at level 2. Hypotheses are filtered down to the lower levels. The more reliable interpretations are found in the lower levels.

When reasonable evidence has been supplied to all of the slots of a hypothesized object, that object becomes established, and slot values that yield the highest confidence for the object are chosen from the possibilities. For each established hypothesis, a subclassification following the "a-kind-of" edges backwards proceeds unless a base class has been reached. Once all unreasonable hypotheses have been rejected, the interpretation process begins again. Interpretation continues until no new hypothesis is generated.

Before the final interpretation of an object can be determined, the confidence computed for a specific class must be spread throughout the whole "a-kind-of" hierarchy. In other words, confirming evidence for one class should have the effect of confirming each class in the path from the root to that class and disconfirming every other class. Similarly, disconfirming evidence for one class should also disconfirm every subclass of that class and confirm every other class. The final interpretation of an object results in that class with the strongest global evidence provided that the evidence is above some preset threshold. The process developed for this system can be found in Kim [5].

Multiresolution Interpretation

In the current system, each lower resolution is computed by averaging the gray levels of 2X2 blocks of pixels at the next higher resolution. Since this construction resembles a tapering pyramid, we refer to higher (lower) resolutions as being lower (higher) levels where the highest resolution is level zero. Each level is processed separately and has the effect of creating hypotheses at the next lower level (higher resolution). This process for three levels is shown in Figure 2 in which heavy arrows represent flow of control and light arrows represent data flow. Each level has as input the images for that and the next lower level, object models specific to the resolution at that level, and hypotheses for that level made by the previously interpreted level.

The appearance of each object at each level is modeled in terms of features extracted from the image taking into account the parts of the object and the other objects that provide support. Thus, for example, an airplane is modeled in terms of regions of the appropriate size and shape that represent fuselage and wings (parts) and shadow (support). The object model at various levels will change as features become more or less prominent. For example, airplane engines are included only in the lower level models.

Interpretation begins at the highest level for which the existence of the objects can be inferred. This inference can be broad, as in "Examine only these areas for the objects," or specific, as in "There is specific evidence for the existence of an object here." Examination at lower levels where more object detail is present is guided by previous interpretations. This is an important feature since without the focusing of attention mechanism, larger portions of the cluttered, high resolution images would be examined. Such examination would waste computational resources and could result in more false alarms. Although interpretations from all the levels can be examined, those resulting at the lower levels will be more reliable.

4. EXPERIMENTS

The system has been exercised on two examples. In the first example, which consists of an image containing a single submarine, three resolutions are processed (levels 0, 1, and 2). The level 2 image (128X128) is in Figure 3. Pairs of parallel lines a known distance apart are used to create initial submarine hypotheses at levels 1 and 2. Submarine hypotheses at levels 0 and 1 are made only by hypotheses at levels 1 and 2, respectively. At levels 1 and 2, evidence for a submarine consists of regions representing submarine area and shadow, lines representing glint, and evidence of a submarine at the next lower level. At the lowest level, submarine support consists of regions representing submarine area, shadow, and tail. Since the submarine tail has small area and may not be visible at high levels, it is used as support at level 0 only.

After applying the Canny edge operator to the original image, two-connected contours (each pixel has at most two neighbors) are extracted and broken at points of high curvature. Line segments are then extracted by fitting lines to the broken contours. Pairs of parallel lines with a minimum length and a specified distance apart are then determined. All of the lines chosen as a member of a parallel line pair for level 2 is in Figure 4. Regions are extracted using an edge-based segmentation (Figure 5) (Perkins [9]). Line pairs and regions for levels 0 and 1 will not be shown. The resulting interpretation of level 2 is in Figure 6a. The regions are both submarine area and shadow. The lines are glint and submarine hypotheses. Two submarines are hypothesized because at this level the lines delineating the right side of the submarine and the left side of the dock are relatively close together. In Figures 6b and 6c, the two submarine hypotheses and their evidence are shown. In each of these images, the outer lines created the submarine hypothesis, and the middle line is the glint line.

The interpretation of the next higher resolution, level 1, is shown in Figure 7a. Again, the regions are submarine area and shadow, and the lines are both glint and submarine hypotheses. At this level, the two hypotheses from level 2 are disambiguated resulting in only one submarine hypothesis. The submarine and its evidence are shown in Figure 7b.

The last interpretation, level 0, is in Figure 8a. The submarine hypothesis is made from the hypothesis at level 1. Glint is not used as evidence at this level; the submarine tail hypotheses are the light, compact regions. The final submarine hypothesis and its evidence at level 0 are shown in Figure 8b.

In the second example, the image contains three airplanes and is analyzed using two levels. The level 1 image (256X256) is in Figure 9. Airplane models at both levels consist of the fuselage which is an elongated, high intensity regions; one or two wings which are also elongated and high intensity regions; and airplane shadows which are elongated and low intensity regions. At the lower resolution, level 1, the airplanes are hypothesized using the fuselage. Support consists of neighboring wings oriented at the correct angle and neighboring shadow regions. Airplane hypotheses at level 1 are responsible for creating hypotheses at level 0.

The regions for detecting airplanes are extracted using morphological operators. By shrinking the bright regions in the original image (replace each pixel by the minimum in its neighborhood) and then growing them back (replace each pixel by the maximum in its neighborhood), small, high intensity regions are eliminated. These regions can be recovered by taking the difference of this image and the original image. Small, low intensity regions can be found in a similar way. Bright regions and dark regions for levels 1 and 2 are shown in Figures 10 and 11, respectively. Given these regions, the images are interpreted, starting with the higher level.

The interpretation of level 1 is in Figure 12a. The darker regions represent both wing and fuselage hypotheses. The fuselage hypotheses created the airplane hypotheses. The bright regions are airplane shadows. Figure 12b shows each of the three airplane hypothesis and their evidence where, due to reproduction, only the shadow regions are clearly apparent. Each of the airplane hypotheses at level 1 create hypotheses at level 0.

The three airplanes found at level 0 are in Figure 13. Since the areas to be analyzed are specified by the hypotheses of level 1, very few regions are considered as fuselage, wing, and airplane shadow. Again, due to reproduction, only the shadow regions are clearly apparent.

In both of these examples, the original object hypotheses are made at the higher levels. These hypotheses are then projected to the lower levels. In the submarine example, the lowest level model contains additional detail, namely, the tail. By analyzing the lower levels, the system is able to disambiguate the two original submarine hypotheses. In the airplane image, only those areas of the lower level near the original hypotheses are considered, thereby making the interpretation process more efficient. Finally, by appropriately choosing feature extraction algorithms, airplane and shadow areas are reliably extracted making interpretation of the poor quality image possible. If, on the other hand, we had chosen the edge-based segmentation technique used for the submarine image, the interpretation would have been difficult, if not impossit !e.

5. CONCLUSIONS AND FUTURE WORK

The multiresolution image interpretation system described in this section takes advantage of the reduction in image clutter and object scale to efficiently and reliably detect objects of interest. The system represents symbolic image descriptions and scene knowledge efficiently and applies the scene knowledge to the symbolic descriptions effectively. All symbolic descriptions, including the hypothesized objects, are represented in the Multiresolution Symbolic Pixel Array which allows uniform treatment of all image-based objects in the system. The system incorporates a semantic network model representation that is general enough to be easily extended to include additional objects. The simple interpretation process adhers to the principle of least commitment in two ways: (1) using the "a-kind-of" relations, object hypotheses occur only if there exist supporting intrinsic feature properties, and (2) final interpretations are not determined until all hypotheses have been made. Finally, the system incorporates both data and model-driven processing.

The system has been exercised on a submarine image and an airplane image. Using the focus of attention mechanism during interpretation, the object hypotheses are disambiguated (e.g., the submarine image) and high resolution images are not fully analyzed (e.g., airplane image). The submarine model is augmented at the highest resolution with additional detail.

Several extensions to the system are possible. Additional interactions between levels, that is, upward, downward, and between more than just neighboring levels, will allow object hypotheses from both lower and higher levels. This is important when object supports (or parts) are found at a lower level without the object structure being found at a higher level, or when support does not appear in immediately adjacent levels.

Currently, a separate object hypothesis is instantiated at each level; more appropriate is the creation of a single hypothesis that represents all the evidence for an object at the various resolutions. Object models will be more robust with the explicit inclusion of such ancillary information as sun angle, range, and physical properties. Automatic model generation from a three-dimensional model and ancillary information will make the interpretation process more robust. A mathematically-based, multiresolution, evidence evaluation scheme still needs to be investigated. Finally, experiments with additional images are clearly necessary in order to evaluate the system more fully. This includes the incorporation of robust object models based on a wider variety of image features, and experiments using more than three levels of resolution.

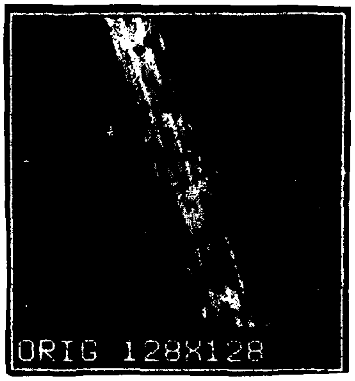


Figure 3. Level 2 submarine image.

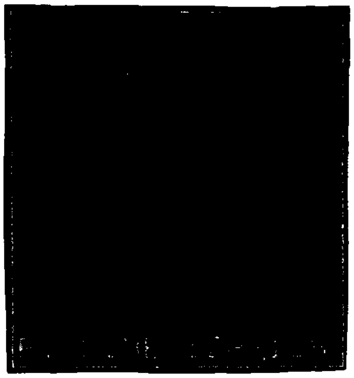


Figure 4. Level 2 parallel line pairs used to create submarine hypotheses.

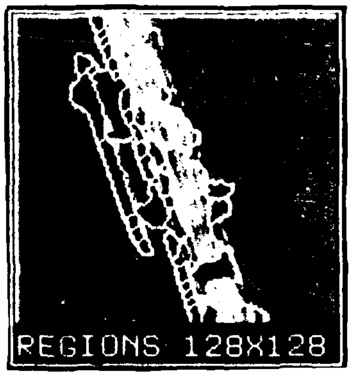


Figure 5. Level 2 regions.

REFERENCES

[1] T. Binford, Survey of model-based image analysis systems, *International Journal of Robotics Research* **1**, 1, pp. 18-64, 1982.

[2] R.A. Brooks, Symbolic reasoning among 3-D models and 2-D images, *Artificial Intelligence* **17**, pp. 285-348, 1981.

[3] B.G. Buchanan and E.H. Shortliffe, eds., *Rule-based Expert Systems: The Mycin Experiments of the Stanford Heuristic Programming Project*, Addison-Wesley, Reading, Massachusetts, 1984.

[4] A.R. Hanson and E.M. Riseman, VISIONS: A computer system for interpreting scenes, pp. 303-333 in *Computer Vision Systems*, eds. A.R. Hanson and E.M. Riseman, Academic Press, Orlando, Florida, 1978.

[5] J.H. Kim, A distributed computational model of plausible classification reasoning, *Proceedings of the Second Conference on Artificial Intelligence Applications*, Miami Beach, Florida, pp. 210-214, 1985.

[6] D.M. McKeown, Jr., W.A. Harvey, Jr., J. McDermott, Rule-based interpretation of aerial imagery, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **7**, pp. 570-585, 1985.

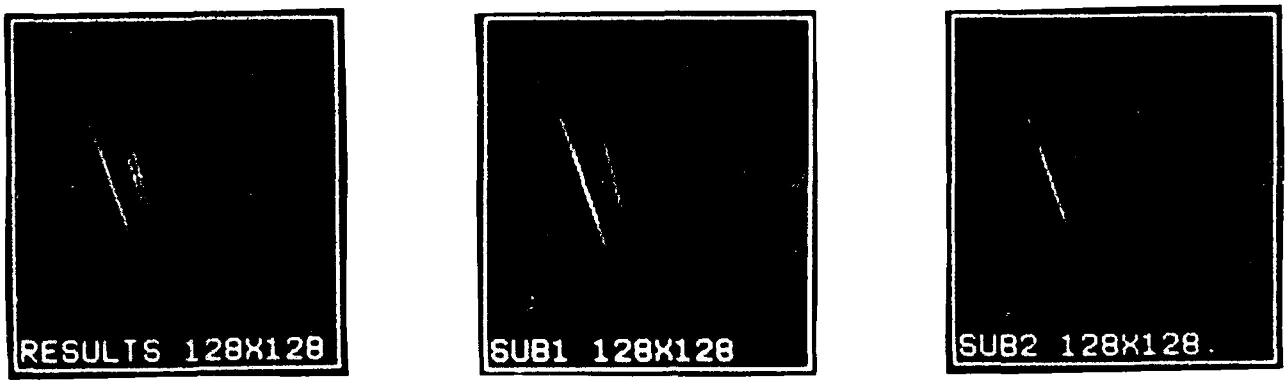
[7] C.F. Neveu, C.R. Dyer, R.T. Chin, Object recognition using Hough pyramids, *Computer Vision, Graphics, and Image Processing* **34**, pp. 52-65, 1986.

[8] D.W. Payton, A symbolic pixel array representatio of spatial knowledge, *Proceedings of the Third IEEE (onference on Computers and Communications*, Phoenix, Arizona, pp. 11-16, 1984.

[9] W.A. Perkins, Area segmentation of images using edge points, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **2**, pp. 8-15, 1980.

[10] T.M. Silberberg, Context dependent target recognition, *Proceedings of the Image Understanding Workshop*, Los Angeles, California, pp. 313-320, 1987.

[11] C.L. Tan and W.N. Martin, A distributed system for analyzing time-varying multiresolution imagery, *Computer Vision, Graphics, and Image Processing* **36**, pp. 162-174, 1986.

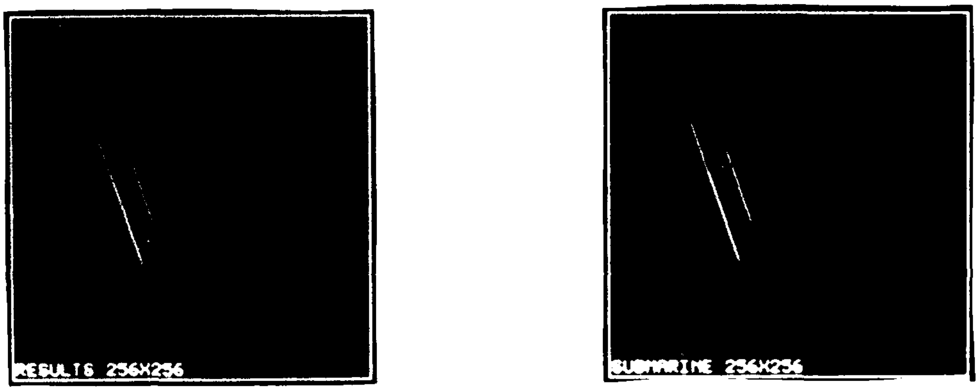


(a) Interpretation of level including all submarine, submarine area, shadow, and glint hypotheses.

(b) First submarine hypothesis and its evidence.

(c) Second submarine hypothesis and its evidence.

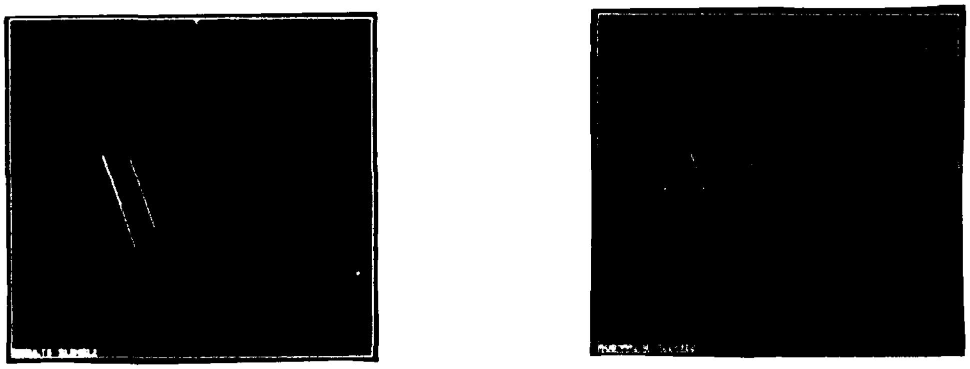
Figure 6.



(a) Interpretation of level 1.

(b) Sub ine hypothesis and evidence.

Figure 7.



(a) Interpretation of level 0.

(b) Submarine hypothesis and evidence.

Figure 8.

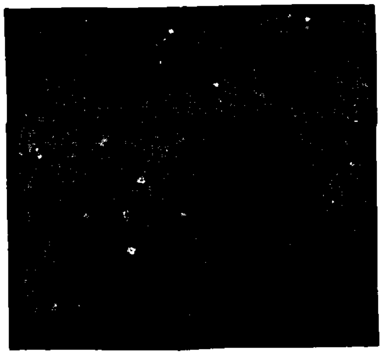
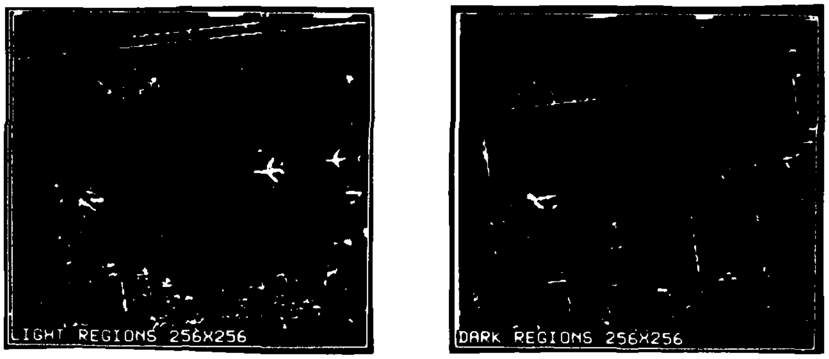


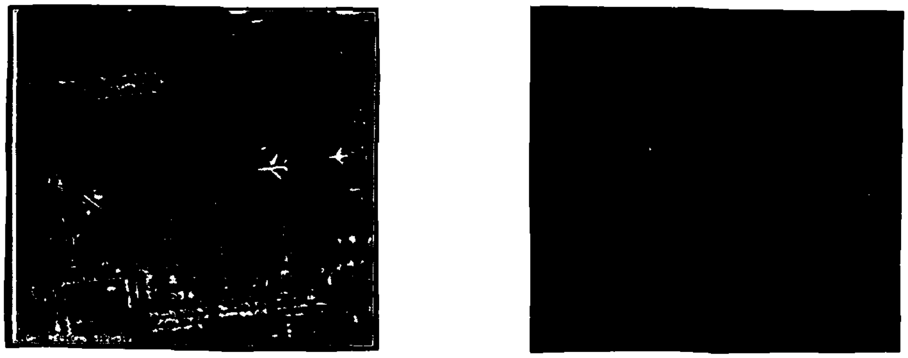
Figure 9. Level 1 airplane image.



(a) Level 1 bright regions.

(b) Level 1 dark regions.

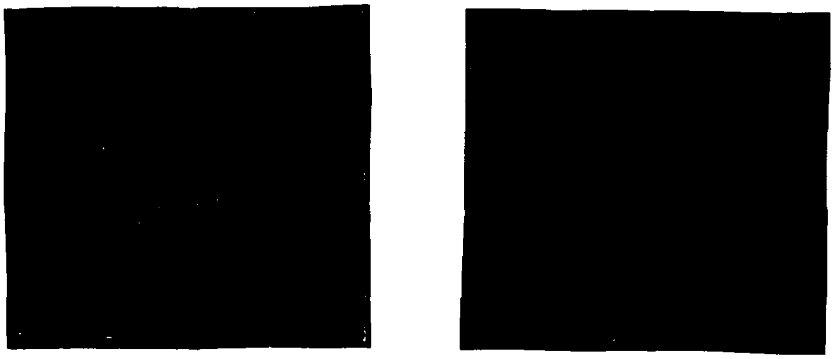
Figure 10.



(a) Level 0 bright regions.

(b) Level 0 dark regions.

Figure 11.



(a) Level 1 fuselage, wing, and shadow hypotheses.

(b) Airplane hypotheses and evidence.

Figure 12.

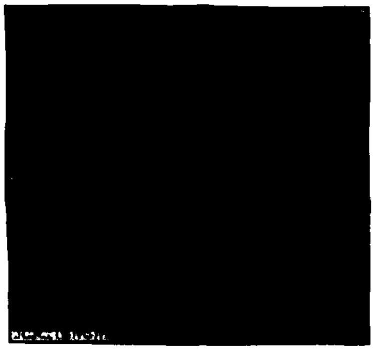


Figure 13. Level 0 airplane hypotheses.

PERCEPTUAL GROUPING FOR THE DETECTION AND DESCRIPTION OF STRUCTURES IN AERIAL IMAGES

Rakesh Mohan and Ramakant Nevatia
Institute for Robotics and Intelligent Systems
Powell Hall 204 - MC 0273
University of Southern California
Los Angeles, CA 90089-0273

Abstract

Vision systems which rely on low-level representations such as edges and regions have difficulty handling complex natural images for high and mid level visual tasks. Representations of structural relationships in the arrangements of primitive image features, as detected by the perceptual organization process, are essential for analyzing complex imagery. We term these representations *collated features.* In the detection of collated features, structural information is utilized to overcome local problems such as noise and poor contrast of real images. The structural information encoded in collated features is essential for such processes as visual reasoning, object segmentation and shape description. Their influence is not limited to the above mentioned "high-level" visual processes as the collated features also actively interact with such "mid-level" processes as stereo and serve as attentional mechanisms for "low-level" feature detection.

We present a vision system that illustrates, for a particular visual domain, the concept of collated features, the process of their detection and their interaction with the diverse visual processes of stereo, monocular and 3D reasoning, shape description and object extraction.

1 Introduction

The ability to extract and describe distinct 3D objects in a complex scene is crucial for an image understanding system. The traditional approaches are edge-based or region-based. In edge-based methods, local edges are detected and then linked into contiguous curves. These curves typically do not give complete boundaries for complex objects and many curves that correspond to texture, surface marking and noise are present. Many attempts have been made to connect these curve segments, using "contour tracing" methods, into meaningful objects. Such techniques have been successful for relatively simple scenes but fail in more complex environments. Region-based methods do give closed regions, by construction, but the regions often do not correspond to the objects in complex environments. The key problem with these segmentation techniques is their myopic nature. The operate locally on the image intensities and do not utilize global information. Thus, these systems fail when faced with local problems such as poor contrast, noise, shadows and occlusions.

One alternative to dealing with these fragmented segmentations is to use "model-based" techniques; a survey can be found in [1]. Model-based techniques usually rely on a priori knowledge of the objects in the scene and predict the appearance of an object to the low-level descriptions that can be extracted from the fragmented segmentation. While impressive results can be obtained for restricted domains, we believe that this approach is too restrictive and not usable for complex scenes and complex tasks.

The alternative we pursue in this work is that of grouping the fragmented low-level descriptions into higher-level groups. Humans are very good at such *perceptual grouping* tasks; we can typically see shapes in arrangements of dots or even poor machine generated edge outputs from even very complex scenes. We believe that such grouping must also be an essential element for machine vision systems if they are to have generality and be able to handle models that are not given very specifically. We further believe that such groupings are useful for high-level processes such as object segmentation and shape description and are likely to be a precursor for even intermediate processes like stereo. Stereo is sometimes believed to give unambiguous data to help in the grouping process, but we show that sometimes the reverse may be more appropriate.

In section 2 we discuss the role of perceptual organization in terms of the representations generated by it and the use of these representations by other visual processes. In section 3 we present a visual domain which we will be using to illustrate the concepts of collated features, their detection and their application in various visual pro-

Supported in part by DARPA, contract # F33615-87-C-1436, monitored by the Air Force Wright Aeronautical Laboratories, DARPA order # 3119, and in part by DMA contract # 800-85-c-0008

cesses. In section 4 we present the technique used to detect collated features. All reasonable feature groupings are initially postulated as collated features. A process of mutual cooperation and competition, outlined in section 5, is then used to select the more promising collations. We show the application of collated features for stereo matching in section 6, and for visual reasoning, object segmentation and shape description in section 7.

We have tested the vision system employing collated features on several stereo pairs of images. We present some collated features detected, and the 3D descriptions generated of the objects located in some of these images.

2 Role of Perceptual Organization

It has been shown that our visual system can immediately detect such feature relationships as collinearity, parallelism, connectivity and repetitive patterns when shown an otherwise randomly distributed set of image elements [2]. This phenomenon is called perceptual organization and has been studied by investigators of psychology and computer vision [2–13] In spite of this active inquiry into the phenomenon of perceptual organization and attempts at implementing some aspects of it, little systematic use of perceptual organization has been made in computer vision systems. The reasons for this lie in the difficulty in detecting the relationships found by perceptual organizations and in designing proper representations and techniques to use these relationships in other visual processes.

Psychophysical experiments show the preference of our visual system for some particular relationships among image features. Collinearity, connectedness, completeness, symmetry, proximity and repetition are among the relationships preferred. The common denominator of these relationships is their structural nature. Thus, the primary purpose of perceptual organization is to make salient the structural interrelationships between image features. We propose that perceptual organization takes primitive image elements typically generated by low-level segmentation processes and generates representations of feature groupings which encode the structural interrelationships between the component elements. We term these representations as *collated features*. Collated features identify those structural relationships that are most common among objects of our visual domain and remain invariant in 2D projections over most viewpoints and represent these relationships in forms suitable for other visual processes. In other words, collated features identify structures (among arrangements of image features) that have high probability of corresponding to object structures (*significance*) and are useful for other visual processes (*utility*).

2.1 Collated Features

Primitive image features, in their raw form, provide insufficient information (explicitly, for in general terms all visual information is embedded in these features) to higher visual processes. The structural information encoded in collated features is crucial to the proper functioning of higher visual processes. This is a major departure from previous vision systems which rely primarily on such image features as edges and regions. The representational power of primitive image features is very low; generic descriptions of them such as compact regions, blue regions etc. can only support very weak inferences. Moreover, such descriptions are completely unsuitable for deriving meaningful descriptions of objects. Some of the deficiences of primitive image elements can be bypassed by model-based vision systems which match precise models of particular objects directly to image features. Even for such systems, it has been found that some preliminary grouping of image features is useful [3,2].

The complex nature of natural images brings forward another inadequacy of primitive image features as representations useful for higher visual processes. In real images the problems of noise, poor contrast and occlusion often result poor edge contours and region segments. Thus some vision systems which rely primarily on primitive image features, and are able to demonstrate reasonable performances on simple scenes imaged under ideal situations, fail misserably when faced with natural images. On the other hand, for detecting collated feature we use structural information from various parts of the scene which result in more robust detection. This indicates that not only are the collated features generated by perceptual organization useful but *crucial* to higher level visual processing.

Collated features are groupings of image elements structurally relevant to objects in the visual domain. They represent groupings of image features which have a high probability of corresponding to object features. Thus collated features "mark" the image features relevant to object extraction. Collated features identify structural relationships among image features which also exist among object features. Furthermore, collated features collect image features that probably correspond to the same object (or part).

Object shapes are described in terms of component shapes. This decomposition of shape description follows a structural basis where individual shape components are "well formed" and have "simpler" individual descriptions than that of the combined shape. The collation of features identifies such individual structures that are useful in describing overall shapes. Collated features also form small local structural descriptions. These can be combined to give global object descriptions, or be transformed to other descriptions more suitable for a particular visual process.

Experiments by Julesz [4] and various stereo algorithms [16–18] may have portrayed stereo as an early stage processing which is performed prior to any feature grouping. However, the difficulty of fusing random dot stereograms, given the absence of ambiguity in matching such patterns (since random patterns are relatively free from repetitions) in contrast to the ease of fusing real images, indicates that the stereo system *does* use more evolved representations in addition to primitive ones. Supporting evidence from this comes from new stereo systems [19–22] which have demonstrated the use of feature groupings in aiding stereo matching. Collated features are more structurally evolved representations than edges, thus there is much less ambiguity in matching collated features than is in matching primitive features. We believe that matching between collated features generates a rough disparity map by indicating match between feature pools and this can guide matching between more precisely located image features such as edges. While collated features can in a similar fashion help other visual matching tasks such as large time interval motion sequences or matching visual representations (such as maps) to images; we will not discuss them here.

By the identification of significant features, collated features can act as *attentional mechanisms* for detailed visual inspection, guiding the detection of features at interesting locations in greater detail.

3 A Visual Domain

We believe that the representations formed by perceptual organization are crucial for high level visual tasks. To demonstrate our view, we have chosen a domain of natural imagery sufficiently complex to be intractable by techniques primarily dependent on low-level representations such as edges and regions, but with the viewpoints and the shape of the objects so restricted that we can get a simple, unambiguous list of structural relationships to be encoded into collated features by the perceptual organization process. This system will illustrate the methodology for the detection and application of collated features and would serve as a framework for a more general system.

The domain of the vision system *CANC* [5] consists of aerial images of suburban scenes. The objects of interest are buildings. The vision system works by first detecting collated features in the images and then employing these collated features in various visual processes to generate descriptions and a three dimensional model of the buildings in the scene.

3.1 Previous Work

A lot of work has been done in computer vision on detecting buildings and other man made structure in aerial images. While a wide variety of techniques have been applied towards this task, a systematic use of perceptual grouping has been lacking. Another interesting observation is that while man made objects have rich geometric structure, little use of this structural information was made in the older systems.

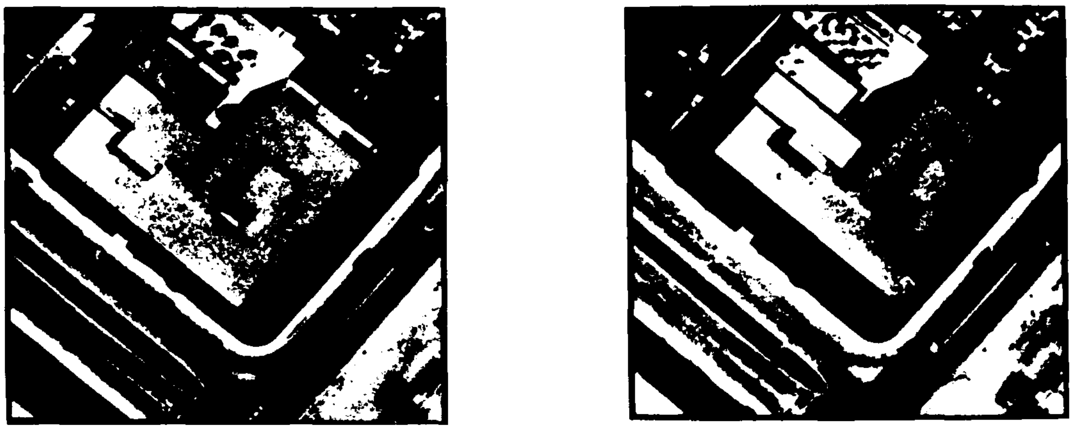
In [6] a region segmenter is used and the relationships between such objects as roads and houses is used to improve detection. In [7] lines and junctions are used as features. Height is obtained from stereo or 2D reasoning to generate a wire frame model which is then completed by hypothesizing likely structures. Contour tracing with some structural guidance as oriented corners and depth from shadow has been used in [26–28]. Fua and Hanson [8] segment the scene into regions, find edges lying on region boundaries and then see if there is evidence of geometric structure among these edges to classify the region as a man-made object. In the VISIONS system [9], region segmentation is the primary technique used and the regions are classified by their shape and spectral properties. SPAM [10] is a map based system which uses region segmentation of aerial imagery. Recently, applications of perceptual grouping to locate features indicating structure have been explored [11].

Most of these systems work in simple scenes, for example rural scenes, where the building roof can be simply segmented (and even identified) from the surround on spectral poperties. The buildings detected have simple shapes (MOSAIC [7] handles complex buildings and urban environments, but makes many errors). Only a few systems compute and use of depth information. None of the systems generate a description of the buildings at the level of shape descriptions of the different wings.

3.2 Problems Specific to this Domain

Fig. 1 shows a stereo pair of images of a building with wings of various heights in a suburban environment. The building is easy for humans to see, even without stereo, but it is in fact very difficult for current vision systems. Fig. 2 shows the line segments detected in the image-pair (Fig. 1). We are still able to see the roof structures of the buildings readily and easily, but the complexity of the task now becomes more apparent. In previous work, assuming a generic model of the building shapes and other knowledge (such as shadows), it has been possible to extract the desired objects in somewhat simpler environments [26–28]. The current scene, however, stretches the limits of such "contour-tracing" like methods.

Many real images, of which aerial imagery of urban scenes is an example, present a multitude of problems which are insurmountable by region and edge based segmentation systems. In the case of buildings, the roofs of their various wings are made of same or similar material.



(a) Right Image

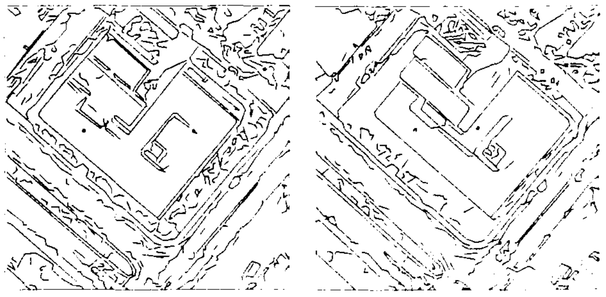
(b) Left Image

Figure 1: Aerial image of a suburban scene.

Therefore, there is often very little contrast between adjoining roofs. Areas adjoining roofs like curbs, parking lots, walkways are often made of materials (for example concrete) similar to that used for the roofs. There are random shaped patches on roofs due to dirt and variation in material. The small structures on the roofs and the objects such as car, and trees adjoining buildings confuse region finders and contour tracers. The shadows and surface markings on the roof cause similar problems.

There are other characteristics of these images which specifically cause problems for contour tracking type systems [26-28]. Roofs have raised borders which sometimes cast shadows on the roof. This results in multiple close parallel edges along the roof boundaries and often these edges are broken and disjoint. At roof corners and at junctions of two roofs, multiple lines meet leading to a number of corners making it difficult to choose a corner for tracking. Roofs cast shadows along their sides and often have objects on the ground near them like grass lots, trees, trucks, pathways etc., which lead to changes of contrast along the roof sides. Thus while tracking one can face reversal in edge direction. Often some structures both on the roof and on the ground are so near the roof that the border edges get merged with the edges of these objects, leading contour trackers off the roofs onto the ground or inside the roof. At junctions it is difficult to decide which path to take. Searching all paths at junctions leads to a combinatorial explosion of paths. It may be difficult to decide on the correct contours since contours may not close because of missing edge information, or more than one closed contours may be generated. Contours may merge roofs or roofs and parts of the ground. Figures 2, 15 and 21 illustrate some of these problems. Note the edges around the roof of the building.

Stereo analysis is also difficult here. The roof tops have little texture and thus little context is available for the matching of their boundaries. In fact, the usual assumption that disparity changes smoothly, is violated in very many places, since the important boundaries largely represent depth discontinuities.



(a) Right Image

(b) Left Image

Figure 2: Linear segments detected in Fig. 1

3.3 Choice of Collated Features

Most buildings such as commercial buildings, residential complexes, warehouses, offices, barracks, airport terminals, railway and bus stations, and school and college buildings can be modeled as being compositions of parallelopiped shapes. In aerial images, building walls get occluded by roofs or other walls and when visible appear heavily foreshortened. Roofs form the only clearly visible parts of the buildings, in such images. The roofs lie parallel to the imaging plane, and the imaging distance is large compared to the size of the roofs, so the view of the roofs is nearly orthogonal.

Roof shapes can be modeled as a combination of rectangles. For example, L, T, H and E shapes are all combinations of rectangles. The important structural relationships among features of objects of this visual domain are those of linearity, parallelism, U-shape-ness and rectangularity. Therefore, it is these feature groupings that would be made significant by perceptual organization. The collated features, which we use, are lines, parallels, Us and rectangles. Note that the collated feature named rectangle is suitable for describing the shape of the objects detected. We have given names such as U and rectangle to these collated features, but these are just convenient nomenclatures for the *structural property* of the collation and does not indicate any recognition or description of the shape at this stage.

In the visual domain we have chosen, the viewpoint is directed towards the ground plane and the choice of collated features reflects this. The problem that some of the structural relationships may change with a change in the direction of the viewpoint does not arise since there is only one allowed viewpoint direction for this domain. It is possible that some other collated features may be formed for this domain, such as groups of regularly spaced parallels (due to roads) and local directional maps [12] (due to textured areas such as grass). We shall not consider such collations here since they are not relevant to buildings.

4 Detection of Collated Features

The detection of collated features works bottom-up from edges, grouping them step by step into more and more geometrically evolved shapes. Initially all reasonable groupings are considered as collated features; this set is then refined by identifying the *significant* collations as more global context from other collated features become available after the initial groupings are made.

The collated features are generated in a recursive manner. First simple collated features are formed by grouping edges on simple geometric relationships. These collations in turn serve as tokens for more complex collated features which represent some geometrical relationship among the component structures. Thus a hierarchy of collated features is created moving from simple collations such as lines up to rectangles. This hierarchy represents an evolution of geometrical organization, elements higher up in the hierarchy reflect more organization and thus have a higher probability of reflecting an actual similar organization in the scene rather than an accidental collection of image elements. The simpler collated features such as lines are less structure specific, since they could belong to a variety of shapes, but as we move up the hierarchy, the more complex collations such as rectangles are more specific because they allow limited structural interpretations of the shape.

The large collated features and their subsidiary collations exchange information in both directions. Simple collated features are used to form the more complex groupings which in turn help the formation of the simpler collated features by use of their more global view. This symbiotic relationship will become more evident in the following subsections.

The collated features provide multiple descriptions of the structure. These descriptions usually differ in the relationships described or the scale. Also alternative groupings, at the same descriptive level, of the features (collated or primitive) exist. Usually a process of selection culls the alternatives as more context from other collated features becomes available, or as other visual processes provide more information, letting only unique collations of the underlying visual tokens to survive. However, there are cases where the visual information is not sufficient to clearly identify the preferred collated features, alternative groupings may survive (as can be seen in some Moire pattern).

We have discussed previously the choice of *lines, parallels, Us* and *rectangles* as the set of collated features. We now describe the process of their detection in detail. The input set of primitive image features consists of *linear segments* and *corners*. We use the "USC LINEAR" system, based on the "Nevatia-Babu" line finder [13] to detect linear segments in the scene. Two types of corners are detected, L and T junctions. We currently do not investigate orthogonal trihedral vertices (OTVs) as few walls are visible, and those that are, appear highly foreshortened and have shadows etc. near them making the OTVs difficult to detect accurately. T junctions for urban aerial imagery do not have, in general, the usual interpretations of occlusion. The buildings have wings which are aligned and nearby objects like roadways, etc. are also aligned to the building sides. Thus in the top view, the sides of two different structures can create T junctions in which the top line belongs to two different objects and is not occluding the stem.

4.1 Lines: Linear Structures

Due to poor contrast, many linear segments along the roof borders get fragmented. Near junctions or close presence of other strong features, these fragments get displaced from the straight line the border lies on, making simple collinearization useless.

A collection of parallel lines bunched along the same linear axis represents the presence of a *linear structure* at a higher granularity level than the edges (for example the boundary of the roof as opposed to the individual lines belonging to its borders). We wish to group closely bunched parallel linear segments since they represent a linear structure of some object, like the border of a roof or the divider on a road.

To detect such groupings of edges, we "fold" the space around each segment onto the segment repeatedly, like pleats in an accordion, collecting the segments from this

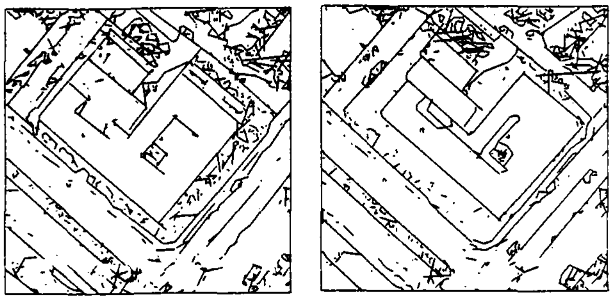

(a) Right Image (b) Left Image

Figure 3: Linear structures obtained from Fig. 2

space which lie parallel to it. This folding process is halted as soon as no new segments are located or when the threshold on the spread about the linear structure is exceeded. Fig. 3 shows the lines obtained from Fig. 2.

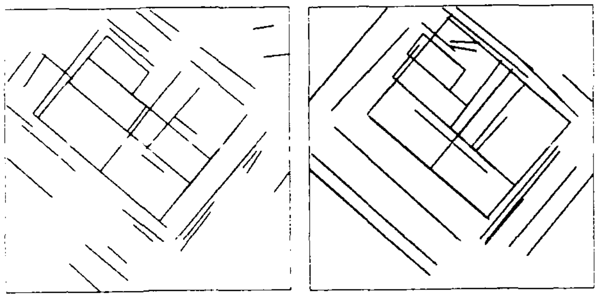

(a) Right Image (b) Left Image

Figure 4: Parallels

4.2 Parallels

For each line obtained above, we find lines that are parallel to it and have a sufficient overlap with it. Man-made structures in urban scenes like building-wings, roads and parking lots are organized in regular grid-like patterns. These structures are all composed of parallel sides. As a consequence, for each significant line-structure detected in the scene, there is not one but numerous lines parallel to it.

The formation of the *parallel* collated features in turn aids the formation of lines. The structure of two parallels lines strongly suggests a complete overlap between the two. If the original lines do not overlap completely, their extensions which will complete the overlap, are considered. These extension are alternate line groupings of the underlying linear segments. Often these new line collations will include new linear segments in the extension, which had not previously been included in the same linear structure as the information (in terms of proximity and collinearity) in the context of the lines alone was not sufficient to trigger the grouping.

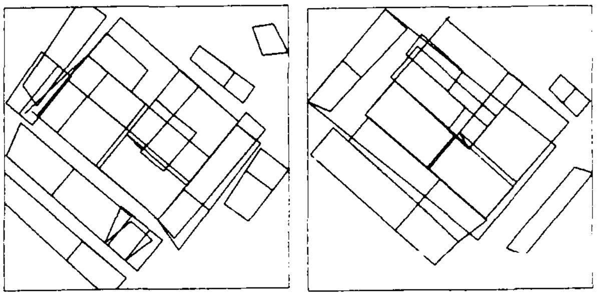

(a) Right Image (b) Left Image

Figure 5: U structures

4.3 U Structures

Each pair of parallel lines evolves into a set of parallels and the ends of their component lines are aligned. A set of parallel lines with aligned ends is a strong indication that there is possibly a line joining those ends to create a U shaped structure.

Thus the presence of a parallel with aligned ends triggers the formation of another collated feature, the U structure. The U collation, or rather the parallel with aligned ends, gives strong suggestion of a line joining the two ends. If an appropriate line joining the aligned ends does not already exist, a new linear collation is created. This new collation may incorporate any existing linear segments or may be "virtual"; again the perception of a complex structure, in this case a U, triggers the formation of a less evolved collated structure, the line.

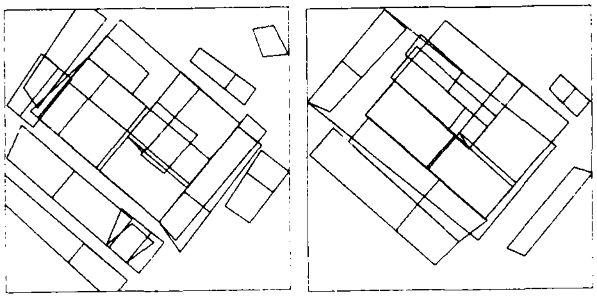

(a) Right Image (b) Left Image

Figure 6: Rectangles

4.4 Rectangles

Each parallel generates two U structures and the Us of a parallel taken together form a rectangle. Us are not the only components of the rectangle, its percept depends on *each of its components*, the lines, the parallels and the Us. While we have chosen to represent the rectangle as one collated feature, a structure of similar complexity may be possibly represented by various collations, each making salient a different structural property such as the various symmetries. For us a single representation is sufficient because of the simplicity of the shapes in our domain and the fact that we shall not make use of the symmetries for any descriptive (or other visual) purposes.

5 Selection of Collated Features

The detection of collated features, where all reasonable groupings among tokens resulted in the formation of collated features, is followed by selection where only the more suitable collated features are retained. The detection and selection processes could proceed simultaneously.

At each level of collated features various collations are in contention as they provide alternative groupings of the underlying tokens. Also some collations may have been formed on weak evidence, evidence that seems too weak when compared to that for other collated features at that level. A selection process has to choose "good" collations, i.e. those which have high probability of corresponding to individual object parts.

The "goodness" of a collated feature depends on how it compares to its alternatives in terms of the support it has from related collations at other levels, and the support or contradiction from its component primitive features and other related image features. A collated feature is not supported just by its component collations but also by the collations it itself is a component of. The later relationship is due to the fact that the percept of a larger structure strengthens that of a smaller component structure, for example as the percept of a U shape strengthens the percept of a line forming the base of the U. Thus a line and the U it belongs to are mutually supportive. In general terms, collated features which are linked by part-of relationships are mutually *supportive* and those that share component collations are mutually *competitive*.

5.1 Constraint Satisfaction Networks

The collated features and the relationships of support and conflict among them naturally define a network with the collations serving as nodes and the relationships as arcs. Our goal is to find the *optimal* feature groupings consistent with the known optical and geometrical constraints [14,15]. Note that all the constraints must be *simultaneously* satisfied to reach global consistency across all levels of the hierarchy.

One parallel technique to solve this problem is relaxation where a cost function associated with the network is minimized. We wish to select the best *consistent* feature groupings, and reject the bad groupings. If we formulate the cost function such that the optimal solution corresponds to its global minima, then the problem of locating the best groupings reduces to that of optimizing the cost of the network given the constraints (defined by the relations) between the collated features and the observed image characteristics. Parallel optimization techniques such as simulated annealing [16], Hopfield networks [17,18], Boltzman machines [19,15,20], probabilistic solutions [21] and connectionist methods [22,19] have been proposed for problems which can be formulated as *Constraint Satisfaction Networks.*

We use a slightly modified version of Hopfield networks to implement the constraint satisfaction network.

5.2 Hopfield Networks

Following the notation convention of Hopfield and Tank [23,24] we describe the behavior of each node in the network by:

$$\begin{aligned} du_i/dt &= -u_i + \sum_{j=1}^{N} T_{ij} V_j + I_i - h_i && (1) \\ V_j &= g(u_i) && (2) \\ h_i &= \textit{resting potential of node}_i && (3) \end{aligned}$$

where T_{ij} is weight on link from node j to node i, I_i is the total input to node i and V_i is the output of node i. The addition of h_i, the resting potential, is useful in adjusting the sensitivity of a neuron by shifting its gain curve. For purposes of analysis of the network, the resting potential may be combined with the input.

When the network has symmetric connections, i.e. $T_{ij} = T_{ji}$, the network, where each element has the above equation of motion, converges to stable states. This property has been shown for more general networks by Hummel and Zucker [25]. Also when the gain function g is high gain (width of the gain curve is narrow), the stable states of the N elements are the local minima of the following cost function with the outputs of the nodes at 0 or 1 [24].

$$E = -\frac{1}{2}\sum_i \sum_j T_{ij} V_i V_j - \sum_i V_i I_i \qquad (4)$$

The signs in the above cost function suggest that if we wish to select mutually supporting collations and reject mutually conflicting collated features, the weights T_{ij} betweei. .upporting hypotheses should be positive and that between conflicting hypotheses should be negative. Those optical and geometrical constraints, which are not expressed purely via the interrelationships between the interpretations should be fed as inputs I_i to the nodes. Again the sign in equation. (1) and (4) shows that supporting evidence should be included as positive input and contradicting evidence as negative input.

In most applications of Hopfield networks, the problem is explicitly stated as the minimization of a particular energy or cost function. If the problem is *naturally* formulated in terms of a cost or energy function, for example the cost of traversing a path in the traveling salesman problem [17] or the energy of a membrane or thin plate in surface interpolation [26], the network is constructed by transforming this cost function to equation 2.4 and selecting the interconnections T_{ij} from this transformation. In other words, when there is an energy function associated with the problem it can be used to specify the interconnections. If the problem is more suitably described as a constraint satisfaction problem, as is our case, with the relationships among the elements specifying the constraints, it is more natural to specify the interconnections from these relationships rather than to formulate an energy function for the network.

5.3 Selection of Constraints

To insure the selection of *perceptually significant* feature groupings in the scene, the choice of weights should reflect the perceptual importance placed on the optical and geometric constraints between the various collated features. The perceptual significance of a collated feature lies in its indication of actual object structure in the scene. For example, while any grouping of parallel lines [11] is indicative of some order in the scene, we are more interested in parallels that actually correspond to individual objects. Therefore, the parallels that have supporting structural evidence such as rectangles are more significant than those that do not.

The relationships between the various collated features are represented by the network in Fig. 7. Collated features which *support* each other are connected via *positively* weighted links (bold lines) while mutually *conflicting* collations are linked via *negatively* weighted links (thin lines).

As Fig. 7 shows, collated features at different levels of the hierarchy which share image features, support each other while groupings of the same level which share image features, conflict with each other. The details of the various nodes and their connections are as follows:

- **Lines:** Lines have two components, *edges* and *gaps*. Edges refer to the intensity discontinuities detected in the image, gaps are sections of the line where no edges were detected. Presence of edges supports the line hypothesis while the presence of gaps weakens it. Lines which share edges are in conflict. Presence of corners at the ends of the line strengthens the percept of the line.
- **Gaps:** Lines aligned with a gap support the line to which the gap belongs, i.e. they weaken the gap hypothesis. On the other hand, lines *crossing across* the gap strengthen the gap hypothesis, i.e. contradict the line percept.

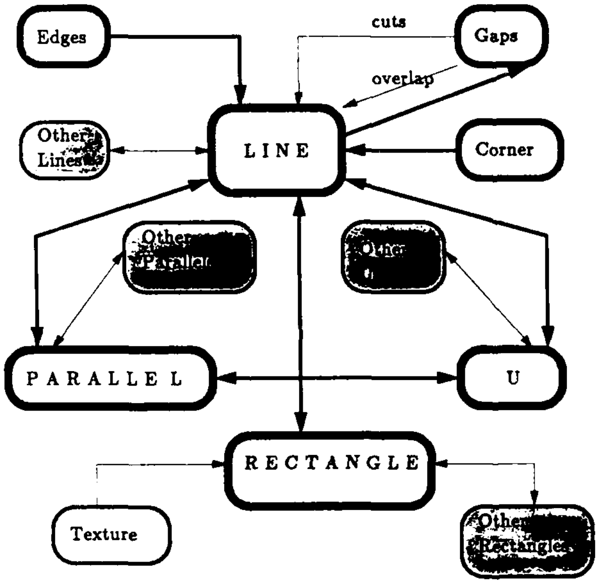


Figure 7: Constraint Satisfaction Network

- **Parallels:** Parallels are supported by their component lines and by the Us and rectangles they are a part of. The various parallels formed from the same line are in competition.
- **Us:** Us are supported by the parallels, the line at the base of the U and the corners at the junctions of the base and the parallels. Us of conflicting parallels are also mutually conflicting. Lines crossing across the base of the U form evidence against it.
- **Rectangles:** The components of a rectangle and the four possible corners are supportive evidence for it. Different rectangles which share the same edges are in conflict, and so are rectangles which overlap each other (i.e. share areas of the image). Surface markings indicating texture elements inside the rectangle further inhibit the rectangle hypothesis.

Each collated feature supports itself (self excitation). In addition to the relationships described above, there is *global competition* among collated features of the same complexity. This is to insure that isolated (i.e. groupings which have no alternative groupings for their component features) but poor grouping hypotheses do not get selected just due to the absence of competition.

The weights on the links range from -1.0 to 1.0. and are in the proportion to the importance of their source collation as supporting or conflicting evidence to their destination collation. The selection of the weights has been random within the confines of the above constraints. We have found that the network is not sensitive to even large changes in the weights if the total amount of evidence ($\sum_{j=1}^{N} T_{ij}V_j + I_j$) arriving at the nodes does not change drastically. The network, in its present form does not have the ability to "self calibrate" i.e. to automatically adjust the sensitivity of a node on the basis of the total amount of information arriving at it. The sensitivity of the nodes can be controlled by using a bias similar to the resting potential of neurons. By controlling the bias we control the amount of positive evidence required by a node to fire it.

While feature groupings at all levels of complexity get selected simultaneously, only the selected rectangles have been displayed in Fig. 8. In our implementation, the weights on the links are not symmetric, so the convergence results for Hopfield networks can not be used. However there is support that the networks can converge under non-symmetric weights [27]. We have found our networks to converge on all our selection of weights within ten iterations.

It may be useful to note here that unlike many other applications of "neural-networks" to computer vision [19], each node represents a high level feature. The features to be represented and the relationships between them have been selected by us. This, however, does not require the network to be made by hand; the nodes and their relationships get automatically formed as a byproduct of the detection process.

6 Applications to Stereo

There is a strong belief in the computer vision community that stereo correspondence precedes any analysis of the structures in the image. This belief started with the experiments by Julesz [4] and has been demonstrated in the design of stereo algorithms using edges or intensity as matching primitives. Julesz's experiment just showed that stereo correspondence can use primitive image elements before any grouping is performed on them; it does not restrict other, more structurally evolved, representations from being used by the stereooptic process. In fact, the difficulty encountered by humans in fusing random dot stereograms indicates that the stereo process is working with a smaller class of representations than it usually does. More recently, stereo systems have shown improved performance by using more structure (than individual edges) [28,29,30,31,32].

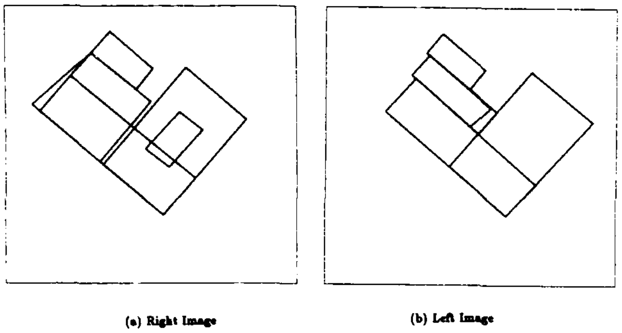
Collated features are rich representations. They encode particular structural relationships at a particular scale of description. Matching collated features thus involves less ambiguity than edges as there are less possible alternatives and more information to judge a match. Also there are usually much less collated structures, at any given representation level than edges. The most probable role of collated features, and one that we employ here, is that correspondence of collated features provides a rough correspondence for their component primitive features, which can then be matched with less ambiguity.

For the vision system *CANC* we use the rectangle collated features to aid stereo matching. For this visual domain, edge and segment based stereo matching algorithms displayed poor performances. The following factors indicate why stereo systems based on simple image features may not perform well in this domain:

1. *Organized nature of the scene.* The are numerous parallel lines since the building-wings, roads, parking lots, etc. are all parallel. Roof borders and their shadows and road markings also give rise to close parallel lines. This leads to many ambiguous matches and it is difficult to resolve among the competing matches.
2. *Absence of texture.* The buildings sides represent areas of high disparity change and there are insufficient markings on the roofs to support match-disparities at roof level while matches giving low disparities get favored due to the preponderance of features on the ground.

The choice of rectangles restricts the possible matches. These can be further constrained by observing that roofs are parallel to the imaging plane. This implies that the disparity along the side of a rectangle, corresponding to a segment of a roof, remains constant. Due to occlusion, however, the disparities of the different sides of a rectangle can all be different with the disparity of the side corresponding to the disparity of the object occluding the rectangle. Since the area bounded by a rectangle is assumed to correspond to a horizontal roof, the rectangle can be assigned the minimum disparity of its sides, assuming the sides with greater heights to belong to occluding objects. In the case of a hole in the roof or in extreme cases of occlusion, the disparity of the area bounded by the rectangle may be different from that of any of its sides. For proper handling of such cases the disparity of the area inside the rectangle should be obtained, but obtaining reliable matches with the small amount of texture commonly found on roofs leads us into issues in stereo matching that we do not wish to address here.

Finding a match between two rectangles corresponds to assigning a unique one-to-one correspondence between the sides of the rectangles. Thus to check if two rectangles match we have to check if sides of the two match in order; i.e. if side *ab* of rectangle *A* matches side *cd* of rectangle *B* then all the sides of *A* on the left (or right) of *ab* must match all the sides of *B* on the left (or right) of *cd*. As mentioned earlier, the sides of the rectangles of interest are parallel to the imaging plane. Therefore, two sides are possible matches if they are nearly parallel and lie within the epipolar window of each other (i.e. if *ab* matches *cd* then at least *c* or *d* should lie within the



(a) Right Image

(b) Left Image

Figure 8: Rectangles selected by CSN

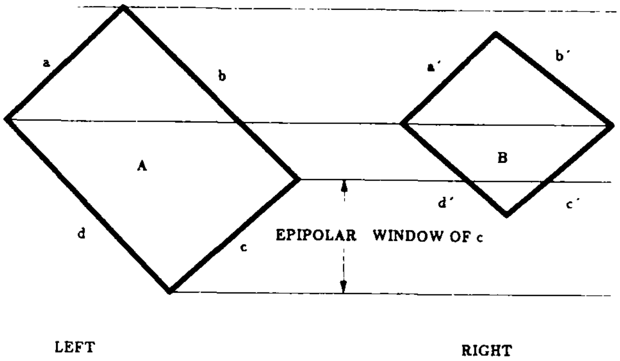
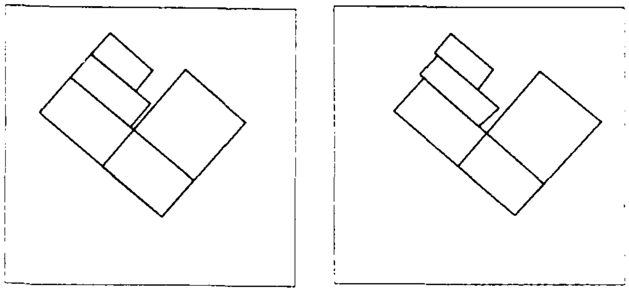


Figure 9: Stereo matching of the rectangle collated feature

interval defined by the epipolar lines of *a* and *b*). The presence of L junctions provide an added constraint that the two sides joined by the junction have the same disparity, whether the sides belong to the area bounded by the rectangle or to an occluding object (which we assume for roofs can only be another roof). If the sides of two rectangles match in order meeting the above constraint then we hypothesize a possible match between the two. Like other stereo matching systems we allow only matches falling within a disparity range reasonable for the stereo pair. To avoid mistaking rectangles corresponding to tennis courts, parking lots and the like, the legal disparity range should start just above ground level. The other end of the interval should be high enough to encompass the tallest buildings in the scene. This estimate need not be exact as possible wrong matches between rectangles result in disproportionate disparities. For our test cases we choose an ad hoc value which was more than twice the disparity of the tallest building in any of the test scenes.

The key problem with general stereo systems is the ambiguity in matching necessitating a mechanism to choose one among many competing matches for each match primitive. For this system we have found the constraints imposed by the structure of the collated feature sufficient to select unique matches for the primitives (rectangles). In the rare case of a rectangle finding more than one match, we choose the match with the least disparity difference between the sides, which is equivalent to prefe-ring the least occluded interpretation.

Stereo can serve as an important visual clue in selecting those collated features which have a very good chance of corresponding to actual object structures, in this case the roofs. Selection of the proper collated features is crucial for this domain as many other objects in the scene such as road segments, parking lots and sidewalks have rectangular structures. Furthermore, these objects are arranged in a regular grid like manner, and some collated features formed reflect the structure in the layout of the scene rather than that of specific objects. In general, objects in a scene are not organized in a regular fashion, and other sources of visual information such as stereo may not be required for aiding the selection process. The rectangle collations which are components of the roof shapes have heights above the ground, and the disparities of their sides lie within ranges reasonable for the stereo pair. The rectangle collated features that meet this criteria are selected out form the rest. These have high probability of belonging to roofs or parts of roofs. Rectangle-groupings so selected are shown in Fig. 9.



(a) Right Image

(b) Left Image

Figure 10: Rectangles matched by stereo

There is a loss of accuracy in the determination of the disparities as a result of the robustness in the detection of the matched primitives. The rectangles are collated features, and are thus primarily structural representations with low positional accuracy. The component lines of the rectangle only represent the structure among the underlying edges, not their positions. For obtaining accurate disparity, matching of more precisely located features, namely the edges, is required. The lines of the rectangles are replaced by the linear segments they represented and these are then matched. There is little ambiguity in matching linear segments at this stage, since the linear segments of a line are matched only to the linear segments of the corresponding matching line found during matching the rectangles. While each line might represent multiple close parallel linear segments, we choose only the innermost edges for match. The rectangles represent areas bounded by the lines and so the inner edges are a sensible representation of the rectangle. This is a convenient approximation and may not be correct for all cases. We are currently working on using sensitive edge detectors on magnified portions of the image in small windows around the lines for precise detection and location of edges. We can consider even weak edges near the noise level of the image, since we have an idea of the direction of the edges and their geometry (straight lines) and an approximate idea of their location.

7 Applications to Shape Description and Object Extraction

We combine the collated features into structures corresponding to the objects. The compositions of the collations provide a basis for the description of the final shape. The compositions themselves identify individual object parts (object extraction). Since our choice for the collated feature corresponding to the rectangle, identifies the same structure as the shape primitive (rectangle) we employ to describe the object shapes, no translation from the shapes of the collated features to the shapes of the description primitives is required.

The task of combining the collated features into shapes has to be mediated by the process of visual reasoning. Since the shapes arising from the combinations correspond to individual objects, some of them may not exhibit the structural regularity of the collated features and thus may not directly correspond to a collated feature but may have to be derived by combining collated features. The visual reasoning carried out currently is primarily monocular, augmented by stereo as needed.

In contrast to previous uses of monocular analysis, we work with more organized structures than lines and junctions. Also T junctions, which are a key element in monocular analysis can not be utilized for this application domain because of the presence of false T junctions due to alignment. As with other phases of processing, the reader will note that the organized na*ure of the primitives used for processing bring more i ɔrmation to the monocular analysis than is available w.h just edge and junction information.

The structural relationships being considered are those of *subsumption* or *inclusion*, *merger-compatibility*, *occlusion* and *incompatibility*. Let the shapes be defined by their boundaries. Consider two structures A and B. The bounding contours of the structures correspond to groupings of the underlying intensity edges. Sections of the boundary thus correspond to grouped edge-contours, single edge-contours or are abstractions generated by the grouping process.

We find the intersections between the boundaries of A and B. These intersections divide the boundary of each structure into *contour-segments*. The contour segments of each structure are then assigned to one of three disjoint sets, one containing segments that lie *outside* the other structure, one containing segments that lie *inside* the other structure and one containing segments *shared* by the other structure. Since the positioning of the boundaries is approximate, allowances have to be made during the computation of these sets to account for these inaccuracies (for example close parallel and overlapping boundaries may be termed "shared").

O_{AB} : Set of contour segments of A outside B
I_{AB} : Set of contour segments of A inside B
S_{AB} : Set of contour segments shared by A and B, $S_{AB} \equiv S_{BA}$
$Edg(X)$: Set of edges of the set of contour segment, X.

Subsumption: If the outside-segment set of shape A is empty and the shared segment set non-empty and the edge-support for segments in the inside-segment set is poor or non-existent, then we say that structure A is subsumed by structure B and can be removed.

$$(O_{AB} = \phi) \wedge (S_{AB} \neq \phi) \wedge (Edg(I_{AB}) < \tau) \Rightarrow B\ subsumes\ A$$

The following relationships are only checked when subsumption is not present.

Occlusion: *If the contour-segments of A inside B have* strong edge support and those of B inside A have weak intensity edge support then we can assume that shape A occludes shape B. Note that this applies even if the rest of the contour-segments of A and B belong to the shared set or outside set.

$$(Edg(I_{AB}) > \tau) \wedge (Edg(I_{BA}) < \tau) \Rightarrow A\ occludes\ B$$

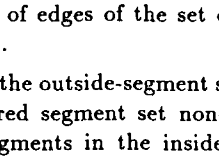


Figure 11: Only possible combination of rectangles in Fig. 10.b

Merger-Compatibility If the segments in the inside-segment and shared-segment set for both shapes A and B have poor edge support then we can conclude that A and B represent segmentation of the one structure into two parts and can thus be merged into one structure. The merger operation is that of union. Note that an implicit assumption has been made that the outside-segment set of A is non-empty since each shape has to have some edge support for it to be formed. The one combination of rectangles resulting from this process on Fig. 10.b is shown in Fig. 11.

$$(Edg(I_{AB}) < \tau) \wedge (Edg(I_{BA}) < \tau) \wedge (Edg(S_{AB}) < \tau) \Rightarrow A \cap B.$$

If on the other hand, the outside-segment set of A is empty and the shared-segment set of A has poor edge support then A and B are merged using the difference operation B - A.

$$(Edg(I_{AB}) > \tau) \wedge (I_{BA} = \phi) \wedge (O_{AB} = \phi) \wedge (Edg(S_{AB}) < \tau)$$

$$\Rightarrow B - A.$$

If stereo information is present, it should be checked that the heights of A and B are compatible since edge support could be lacking in shared-segments of adjoining objects of similar surface properties due to the absence of contrast.

Unrelated: If A and B have null inside-segment sets and null shared-segment sets then they are unrelated. If A and B have non empty shared segment set (and null inside-segment sets) but the shared segments have good edge support then A and B are still unrelated (though adjoining).

$$(I_{AB} = \phi) \wedge (I_{BA} = \phi) \wedge ((S_{AB} = \phi) \vee (Edg(S_{AB}) > \tau))$$

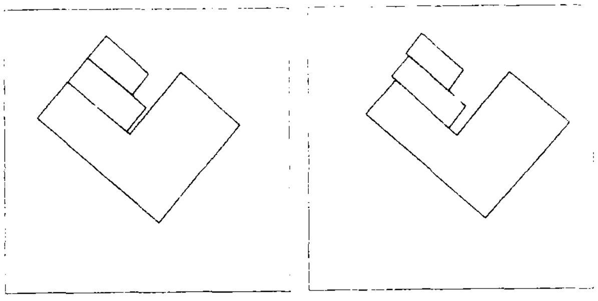
$$\Rightarrow \textit{A and B are unrelated.}$$

Incompatible If A and B have non-empty inside-segment sets and the elements of the inside-segments of *both* A and B have strong edge support then at least one of A or B is a wrong structural grouping and must be deleted.

$$(I_{AB} \neq \phi) \wedge (I_{BA} \neq \phi) \wedge (EDdgI_{AB} > \tau) \wedge (Edg(I_{AB}) > \tau))$$
$$\Rightarrow \textit{A and B are incompatible.}$$

The decision of which structure is erroneous is difficult to make in the context of just two structures, one could possibly retain the structure with more edge and corner support inside the other. If A and/or B conflict with other structures then the one with the most conflicts can be deleted first, and so on. In the case of a tie where we are left with a pair of mutually conflicting structures, other information such as stereo could be used to resolve conflicts.

Our current system reports on the conflicts but does not resolve them. In our test cases we had only one case of conflicting structures, and there the decision was easy to make (manually) as one of the structures was not only conflicting with a number of other structures but was also being "occluded" by a structure of lower height (as reported by stereo) than itself.



(a) Right Image (b) Left Image

Figure 12: Final combination of rectangles (corresponding to roofs)

Starting from the rectangles selected from the previous stages, we perform the above analysis on all pairs of rectangles, first removing subsumed structures and then forming new structures on any possible mergers of rectangles. The process is recursively applied to the new structures along with the original structures from the previous step until no new structures are formed. During this combination, duplication of the structures is possible, but it is trivial to detect duplicate structures since they have exactly the same component rectangles.

The geometrical relationships among the shape primitives (rectangles) and their combinations form a graph which is a structural description of the objects in the scene in terms of the primitive. Structures in the graph which are not marked as subsumed, merged or incompatible are selected as the top level descriptions of the objects or object parts visible in the scene (roofs for our image domain).

The final structures are assigned heights from the disparity information previously obtained by stereo. The buildings are modeled by drawing walls straight down from the from the sides of the roofs to the piane below. be it of another roof or the ground. The resulting model is displayed in Fig. 13.

Results on two more image pairs are shown in the following figures.

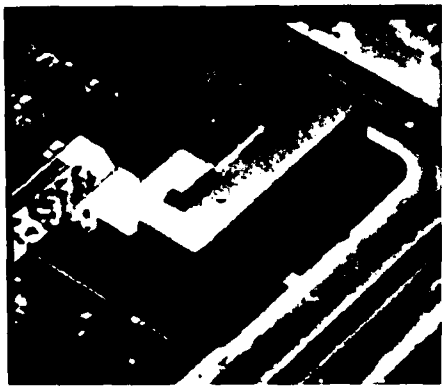
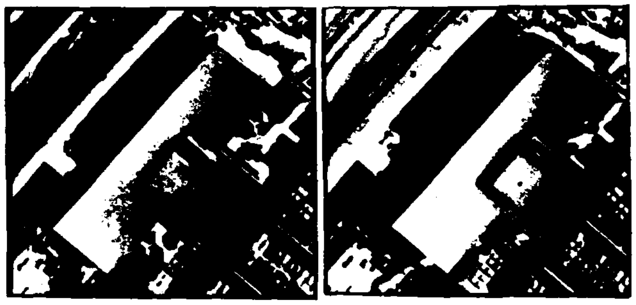


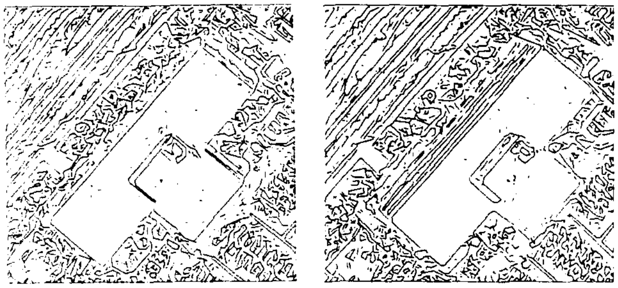
Figure 13: Rendered view of 3D model of building detected



(a) Right Image

(b) Left Image

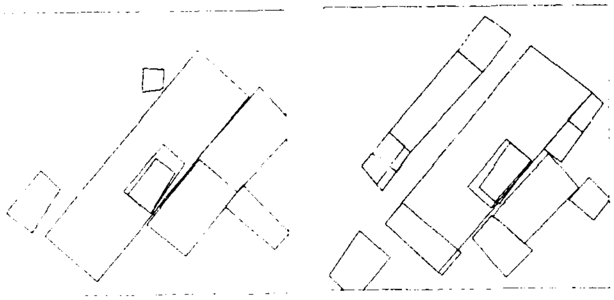
Figure 14: Aerial Image II



(a) Right Image

(b) Left Image

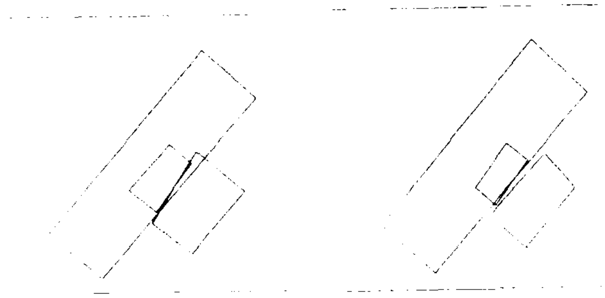
Figure 15: Linear segments detected in Aerial Image II



(a) Right Image

(b) Left Image

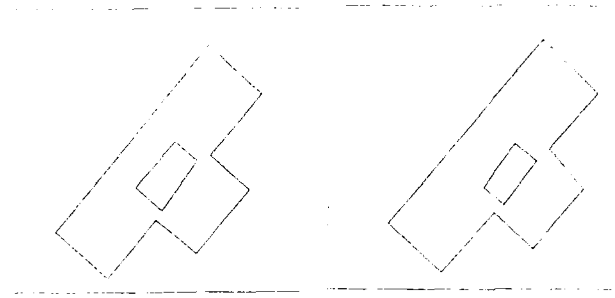
Figure 16: Rectangles selected by CSN



(a) Right Image

(b) Left Image

Figure 17: Rectangles matched by stereo



(a) Right Image

(b) Left Image

Figure 18: Final combination of rectangles (corresponding to roofs)

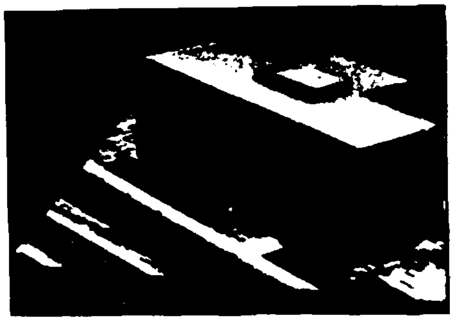
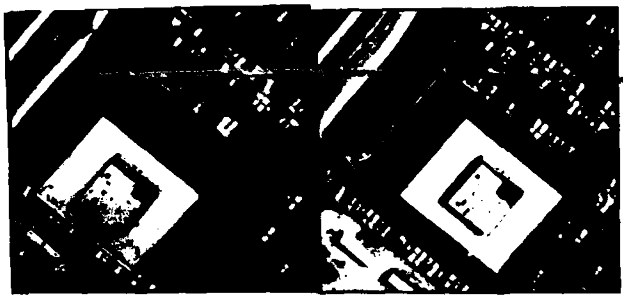


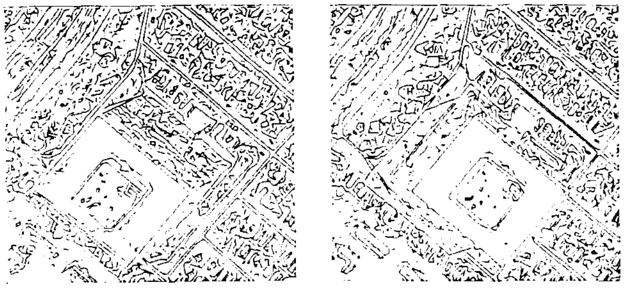
Figure 19: Rendered view of building for Aerial Image II



(a) Right Image

(b) Left Image

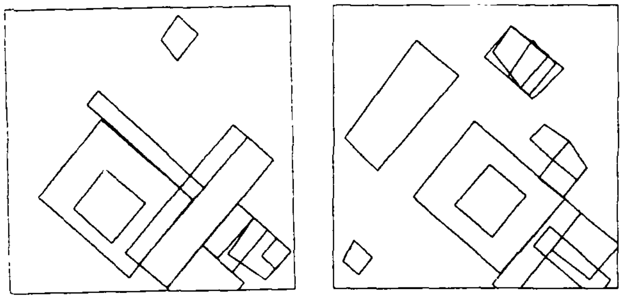
Figure 20: Aerial Image III



(a) Right Image

(b) Left Image

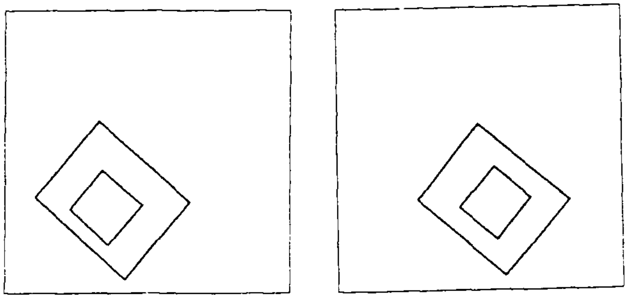
Figure 21: Linear segments detected in Aerial Image III



(a) Right Image

(b) Left Image

Figure 22: Rectangles selected by CSN



(a) Right Image

(b) Left Image

Figure 23: Rectangles matched by stereo (none of the rectangles merge, the final shapes formed, corresponding to the roofs is the same)

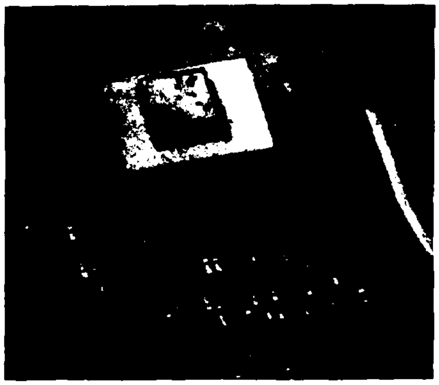


Figure 24: Rendered view of building for Aerial Image III

8 Conclusions

We have proposed *collated features* as the representations computed by the process of perceptual organization applied to the primitive image elements. These collations represent structural relationships between the arrangement of their tokens. We have identified the structural relationships so represented, in terms of their significance for the shapes in our visual domain and the utility to other visual processes. Further we have shown that collated features are useful for stereo and the generation of shape descriptions and object segmentation.

We have illustrated our theory with a working vision system which works on real images from a restricted shape domain. The fact that other techniques such as model matching are not applicable, and that attempts at using techniques based on primitive image elements (such as contour tracing) have been unsuccessful on this domain, supports the need and usefulness of collated features.

References

[1] T.O. Binford. Survey of model based image analysis systems. *International Journal of Robotics Research*, 1(1), 1982.

[2] D.G. Lowe. *Perceptual Organization and Visual Recognition*. Kulwer Academic Press, Hingham, MA, 1985.

[3] R.A. Brooks. Symbolic reasoning among 3-d models and 2-d images. *Artificial Intelligence*, 17(1-3):285–348, 1981.

[4] B. Julesz. *Foundations of Cyclopian Vision*. University of Chicago Press, Chicago, 1971.

[5] R. Mohan and R. Nevatia. Perceptual grouping with applications to 3d shape extraction. In *Proceedings of the IEEE Computer Society Workshop on Computer Vision*, Miami, Florida, December 1987.

[6] Takashi Matsuyama and Vincent Hwang. SIGMA: a framework for image understanding: integration of bottom-up and top-down analyses. In *Proceeding of the Ninth International Joint Conference on Artificial Intelligence*, Los Angeles, California, August 1985.

[7] M. Herman and T. Kanade. The 3d MOSAIC scene understanding system: incremental reconstruction of 3d scenes from complex images. In *Proceedings of the DARPA Image Understanding Workshop*, pages 137–148, 1984.

[8] Pascal Fua and Andrew J. Hanson. Using generic geometric models for intelligent shape extraction. In *Proceedings of the DARPA Image Understanding Workshop*, Los Angeles, CA, Feburary 1987.

[9] A. Hanson and R.E. Riseman. *VISIONS: A Computer System for Interpreting Scenes*. Academic Press, New York, 1978.

[10] D.M. McKeown, W.A. Harvey, and J. McDermott. Rule-based interpretation of aerial imagery. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 7(5):570–585, 1985.

[11] G. Reynolds and J.R. Beveridge. Searching for geometric structure in images of natural scenes. In *Proceedings of the DARPA Image Understanding Workshop*, Los Angeles, C.A., Feburary 1987.

[12] Steven W. Zucker. *Human and Machine Vision*, chapter Computational and Psychophysical Experiments in Grouping: Early Orientation Selection, pages 545–567. Academic Press, New York, NY, 1983.

[13] R. Nevatia and K.R. Babu. Linear feature extraction and description. *Computer Vision, Graphics and Image Processing*, 13:257–269, 1980.

[14] D.H. Ballard, G.E. Hinton, and T.J. Sejnowski. Parallel visual computation. *Nature*, 306:21–26, November 1983.

[15] S.E. Fahlman and G.E. Hinton. Connectionist architectures for artificial intelligence. *IEEE Computer*, 100–109, January 1987.

[16] S. Kirkpatrick, C.D. Gelatt, and M.P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

[17] J.J. Hopfield and D.W. Tank. "Neural" computation of descisions in optimization problems. *Biological Cybernetics*, 52:141–152, 1985.

[18] J.J. Hopfiled and D.W. Tank. Computing with neural circuits: a model. *Science*, 233:625–633, 1986.

[19] D.E. Rumelhart, McClelland, and the PDP Research Group. *Parallel Distributed Processing: Explorations in the Microstructures of Computing.* M.I.T. Press, 1986.

[20] S.E. Fahlman, G.E. Hinton, and T.J. Sejnowski. Massively parallel architectures for AI: NETL, Thistle, and Boltzman machines. In *Proceedings, National Conference on A.I.*, American Association for A.I., William Kafman, Inc., Menlo Park, C.A., 1983.

[21] S. Geman and D. Geman. Stochastic relaxtation, gibbs distributions, and the bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(6):721–741, November 1984.

[22] J.A. Feldman and D.H. Ballard. Connectionist models and their properties. *Cognitive Science*, 6:205–254, 1982.

[23] J.J. Hopfield and D.W. Tank. Neural networks and physical systems with emergent collective computational abilities. *Proceedings, National Academy of Science, USA*, 79:2554–2558, April 1982.

[24] J.J. Hopfield. Neurons with graded response have collective computational properties like those of two-state neurons. *Proceedings National Academy of Science, USA*, 81:3088–3092, May 1984.

[25] R.A. Hummel and S.W. Zucker. On the foundations of relaxation labeling process. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 5(3):267–287, May 1983.

[26] C. Koch, J. Marroquin, and A. Yuille. *Analog "Neuronal" Networks in Early Vision.* Technical Report A.I. Memo 751, Massachusetts Institute of Technology Artificial Intelligence Laboratory, June 1985.

[27] G.A. Carpenter, M.A. Cohen, S. Grossberg, T. Kohonen, E. Oja, G. Palm, J.J. Hopfield, and D.W. Tank. Technical comments: computing with neural networks. *Science*, 235, March 1987.

[28] R. Mohan, G. Medioni, and R. Nevatia. Stereo error detection, correction and evaluation. In *Proceedings of the First International Conference on Computer Vision*, IEEE, London, June 1987.

[29] Y. Ohta and T. Kanade. Stereo by intra and inter-scanline searching using dynamic programming. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, March 1983.

[30] G. Medioni and R. Nevatia. Segment-based stereo matching. *Computer Graphics and Image Processing*, 31:2–18, 1985.

[31] S.D. Cochran. Steps towards accurate stereo correspondence. In *Proceedings of the DARPA Image Understanding Workshop*, pages 777–791, Morgan Kaufmann Publishers, Inc., Los Angeles, February 1987.

[32] H.S. Lim and T.O. Binford. Stereo correspondence: a hierarchical approach. In *Proceedings of the DARPA Image Understanding Workshop*, pages 234–241, Morgan Kaufmann Publishers, Inc., Los Angeles, February 1987.

Dynamic Model Matching For Target Recognition From a Mobile Platform*

Hatem Nasr and Bir Bhanu

Honeywell Systems and Research Center
3660 Technology Drive
Minneapolis, Minnesota 55418

ABSTRACT

A novel technique called *dynamic model matching* (DMM) is presented for target recognition from a moving platform such as an autonomous combat vehicle. The DMM technique overcomes major limitations in present model-based target recognition techniques that use a single, static target model, and therefore cannot account for continuous changes in the target's appearance caused by varying range and perspective. DMM addresses this problem by combining a moving camera model, 3-D object models, spatial models, and expected range and perspective to generate multiple 2-D image models for matching. DMM also generates recognition strategies that can emphasize different object features at varying ranges. DMM operates within a larger system for landmark recognition based on the perception, reasoning, action, and expectation paradigm called PREACTE. Results are presented on a number of test sites using color video data obtained from the autonomous land vehicle.

1. INTRODUCTION

Target recognition from a mobile platform such as an autonomous combat vehicle in outdoor scenarios presents one of the most challenging problems of the machine vision community. It requires the ability to recognize targets from varying ranges and perspectives under changing environmental conditions. Earlier approaches emphasized the need for rotation-invariant and range-independent target models.[1,2] It was soon evident, however, that these models are weak because they have few parameters and cannot adequately handle different aspects at different ranges. Weak segmentation methods further aggravate the recognition problem.

Landmark recognition is a typical application of target recognition for autonomous vehicles. It is used to update the land navigation system that accrues a significant amount of error after the vehicle traverses long distances, which is typically the case in surveillance search and rescue missions. The vision system of the autonomous vehicle is required to recognize the landmarks as the vehicle approaches from the road or on terrain.

We have developed an expectation-driven, knowledge-based landmark recognition system called PREACTE that uses a priori, map and landmark knowledge, spatial reasoning, and a novel dynamic model matching (DMM) technique. PREACTE's mission is to predict and recognize landmarks and targets as the vehicle approaches them from different perspective angles at varying ranges. Once the landmarks have been recognized, they are associated with specific map coordinates, which are then compared to the land navigation system readings, and subsequent corrections are made. Landmarks of interest include buildings, gates, poles, and other man-made objects.

DMM departs from previous model-based and prediction-based vision systems[3,4] by addressing the following requirements:

- Target models are dynamic.
- Different targets require different representation and modeling techniques.
- Single targets require hybrid models.
- At different ranges, different matching and recognition plans need to be performed.

DMM generates and matches target landmark and map site descriptions dynamically based on different ranges and perspectives.

These descriptions are a collection of spatial, feature, geometric, and semantic models. From a given (or approximated) range and view angle, and using a priori map information, 3-D landmark models, and the camera model, PREACTE generates predictions about the individual landmark location in the 2-D image. The parameters of all models are a function of range and view angle. As the vehicle approaches the expected landmark, the image content changes, which in turn requires updating the search and match strategies. Landmark recognition in this framework is divided into three stages: detection, recognition, and verification. At far ranges, only "detection" of distinguishing landmark features is possible, whereas at close ranges, recognition and verification are more feasible, since more details of objects are observable.

In the following sections we present a brief description of PREACTE, details of DMM, and show results on real imagery. More details on PREACTE can be found in Nasr and Bhanu.[5,6]

2. CONCEPTUAL APPROACH

The task of visual landmark recognition in the autonomous combat vehicle scenario can be categorized as uninformed or informed. In the uninformed case, given a map representation, the vision system attempts to attach specific landmark labels to image objects of an arbitrary observed scene and infers the location of the vehicle on the map (world). In this case, image to map registration and spatial or topological information about the observed objects is typically used to infer their identity and the location of the robot on the map as a result. In the informed case, while the task is the same as before, there is a priori knowledge (with a certain level of certainty) of the past location of the robot on the map and its velocity. It is the informed case that is of interest in this paper.

Figure 1 illustrates the overall approach to PREACTE's landmark recognition task. It is a top-down, expectation-driven approach, whereby an expected site model (ESM) on the map is generated based on extensive domain-dependent knowledge of the current (or projected) location of the vehicle on the map and its velocity. The ESM contains models of the expected map site and its landmarks. These models provide the hypotheses to be verified by a sequence of images acquired at predicted times t, given the velocity of the vehicle and the distance between the current site and the predicted one. Figure 2 illustrates this concept. As shown, map site models introduce spatial constraints on the locations and distributions of landmarks, using a "road" model as a reference. Spatial constraints greatly reduce the search space while attempting to find a correspondence between the image regions and a model. This mapping is usually many-to-one in complex outdoor scenes because of imperfect segmentation.

The ESM is dynamic in the sense that the expectations and descriptions of different landmarks are based on different ranges and view angles. Multiple and hybrid landmark models are used to generate landmark descriptions as the robot approaches a landmark, leading to multiple model/image matching steps. This is what is referred to as dynamic model matching (DMM). The landmark descriptions are based on spatial, feature, geometric, and semantic models. There are two types of expectations: range dependent and range independent. Range-dependent

*This research was supported by DARPA under Contract No. DACA76-86-C-0017.

expectations are landmark features such as size, length, width, volume, etc. Range-independent ones include color, perimeter squared over area, length over width, shape, etc.

Different landmarks require different strategies and plans for detection and recognition at different ranges. For example, a yellow gate has a distinctive color feature that can be used to cue the landmark recognition process and reduce the search space. A telephone pole, on the other hand, requires the emphasis of the length/width feature.

In PREACTE, given the vehicle position in the map and an acquired image, PREACTE performs the following steps:

1. Generate 2-D descriptions from 3-D models for each landmark expected in the image.

2. Find the focus of attention areas (FOAAs) in the 2-D image for each expected landmark.

3. Generate the recognition plan to search for each landmark, which includes what features will be used for each landmark and at what range.

4. Generate the ESM at that range and aspect angle.

5. Search for regions in the FOAA of the segmented image that best match the features in the model.

6. Search for lines in the FOAA in the line image that best match :he lines generated from the 3-D geometric model (this step is performed at close ranges where details can be observed).

7. Match expected landmark features with region attributes, and compute matching confidences for all landmarks.

8. Correct the approximated range by using the size differences of the suspected landmark in the current and previous frames.

9. Compute the uncertainty about the map site location based on the previous and current matching results.

2.1. MAP/LANDMARK KNOWLEDGE BASE

Extensive map knowledge and landmark models are fundamental to this recognition task. Our map representation relies heavily on declarative and explicit knowledge instead of procedural methods on relational databases. The map is represented as a quadtree, which in turn is represented in a hierarchical relational network. All map primitives are represented in a schema structure. The map dimensions are characterized by their cartographic coordinates. This schema representation provides an object-oriented computational environment that supports the inheritance of properties by different map primitives and allows modular and flexible means for searching the map knowledge base. The map sites between which the vehicle traverses have been surveyed and characterized by site numbers. A large database of information is available about these sites. This includes approximate latitude, longitude, elevation, distance between sites, terrain descriptions, landmark labels contained in a site, etc. Such site information is represented in a SITE schema, with corresponding slots. Slots names include HAS_LANDMARKS, NEXT_SITE, LOCATION, etc.

Each map site that contains landmarks of interest has an explicitly stored spatial model, which describes in 3-D the location of the landmarks relative to the road and to each other. By using a detailed camera model, range, and azimuth angle, we can generate 2-D views of the landmarks as shown in Figure 3. These views contain symbolic and numeric descriptions of the landmarks and their parts.

Given a priori knowledge of the vehicle's current location on the map space and its velocity, it is possible to predict the upcoming site that will be traversed through the explicit representation of map knowledge. The ESM contains information about the predicted (x,y) location of a given landmark and its associated FOAA, which is an expanded area around the predicted location of the object.

2.2. OBJECT MODELING

Landmark predictions are based on stored map information, object models, and the camera model. Each landmark has a hybrid model that includes spatial, feature, geometric, and semantic information. Figure 4 illustrates this hybrid model representation for a yellow gate; this model also includes:

- Map location
- Expected (x,y) location in the image
- Location with respect to the road (i.e., left or right) and approximate distance
- Location in 3-D

The feature-based model includes information about local features, such as color, texture, intensity, size, length, width, shape, elongation, perimeter squared over area, linearity, etc. The values of most of the range-dependent features, such as the size, length, width, etc., are obtained from the generated geometric model at that given range and azimuth angle. Range-independent feature values are obtained from visual observations and training data. Different parts of the yellow gate are represented in a semantic network.

3. DYNAMIC MODEL MATCHING

Each landmark has a number of dynamic models, as shown in Figure 1. The predicted landmark appearance is a function of the estimated range and view angle to the object. The range and view angle are initially estimated from prior locations of the vehicle, map information, and velocity; they can be corrected based on recognition results. The landmark recognition task is performed dynamically at a sampled clock rate. Different geometric models are used for different landmarks; for example, telephone poles can be best represented as generalized cylinders, whereby buildings are better represented as wire frames. The different representations require the extraction of different image features.

There are three basic steps to the landmark recognition process after generating the prediction of the next expected site and its associated landmarks. These are 1) landmark detection, 2) landmark recognition, and 3) map site verification and landmark position update on the map. At each stage, different sets of features are used.

Detection is a focus-of-attention stage; it occurs at ranges, say, greater than 35m. Very few details of landmarks (such as structure) can be observed; only dominant characteristics can be observed, such as color, size, elongation, straight lines, etc. From the map knowledge base, spatial information can be extracted, such as position of the landmarks with respect to the road (left or right) and position (in a 2-D image) with respect to each other (above, below, or between). So, using spatial knowledge abstracted in terms of spatial models and some dominant feature models, landmarks can be detected, but not recognized with a relatively high degree of confidence. However, this varies from one landmark to another; because some landmarks are larger than others, it may be possible to recognize them at such distances.

The second step, landmark recognition, occurs at closer ranges, say, 35 to 10m. At these ranges, most objects show more details and structure. Segmentation is more reliable, which makes it possible to extract lines and vertices. This in turn makes it possible to use detailed geometric models based on representations, such as generalized cylinders, wire frames, and winged edge, depending on the landmarks. Nevertheless, feature- and spatial-based information is still used prior to matching the geometric model to image content, because it greatly reduces the search space. We should note here that the feature and spatial models used in the first step are updated, because obviously the landmarks are perceived differently in the 2-D image at short ranges.

The third step is a verification stage that occurs at very close ranges. At this stage, PREACTE confirms or denies the existence of the landmarks and the map site location to the vehicle. Since subparts can be identified at close ranges for some landmarks, semantic models can be used to produce a higher degree of confidence in the recognition process. Some landmarks may partly disappear from the field of view (FOV) at this range. This information about the potential disappearance of objects from the FOV is obtained from the 3-D model of the landmark, the camera model, and the range.

Recognition plans are explicitly stated in the landmark model for different ranges, as shown below:

```
(defvar yellow-gate
  (make-instance    'object
    :name           'yellow-gate
    :parts          (list y-g-west-wing  y-g-east-wing)
    :geo-location   '(392967.4  1050687.7)
    :plan           '((40 15 detection) (15 8 recognition) (8 0
                      verification))
    :detection      '(color)
    :recognition    '(color length width area p2_over_area shape)
    :verification   '(color length width area p2_over_area  shape
                      lines) ))
```

Once the FOAA for a landmark is determined from the predicted model, all regions from the segmented image are matched against the landmark. More details on this matching technique can be found in Nasr and Bhanu.[5,6]

We compute the uncertainty U_S at each map site location in the following manner:

$$U_s = (U_{s-1} + \alpha D) * \prod_{i=1}^{m} \frac{0.5}{E(1_i)_{max}}$$

where U_S is the uncertainty at site s, U_{S-1} is the uncertainty at the previous site, L is the average accumulated error or uncertainty per kilometer of the vehicle navigation system, α is the number of kilometers traveled between the previous and the current site, and $E(l_i)_S$ is the evidence accumulated about landmark l_i at site s. U_S has a minimum value of zero, which indicates the lowest uncertainty and is the value at the starting point. The upper limit of U_S can be controlled by a threshold value and a normalization factor.

4. RESULTS

PREACTE and DMM were tested on a number of images collected by the vehicle. The PREACTE system was implemented on the Symbolics 3670. The image processing software was implemented in C on the VAX 11/750, and the image data were collected at 30 frames/sec. In this test, the robot started at map site 105 and headed south at 10 km/hr (see Figure 5). The objective of the test was to predict and recognize landmarks that were close to the road over a sequence of frames. Figures 6 through 20 show different stages of landmark recognition at different map locations. The figures show dynamic models generated by PREACTE at varying ranges. They also show how PREACTE changes recognition strategies at different stages of detection, recognition, and verification. In addition, the figures show the computed site uncertainty at each stage. The site uncertainty fluctuates depending on the landmark recognition results and the distance the vehicle has traveled.

In the future, we will extend this approach to the general situation in which the robot may be traveling through terrain and must determine precisely where it is on the map by using landmark recognition.

REFERENCES

1. P.J. Besl and R.C. Jain, "Three-Dimensional Object Recognition," ACM Computing Surveys, Vol. 17, No. 1, March 1985, pp. 75-145.

2. B. Bhanu, Editor, "CAD-Based Robot Vision," *IEEE Computer*, Special Issue on CAD-Based Robot Vision, August 1987.

3. R.T. Chin and C.R. Dyer, "Model-Based Recognition in Robot Vision," ACM Computing Surveys, Vol. 18, No. 1, March 1986, pp. 67-108.

4. R. Horaud and T. Skordas, "Model-Based Strategy Planning for Recognizing Partially Occluded Parts," *IEEE Computer*, Special Issue on CAD-Based Robot Vision, August 1987, pp. 58-64.

5. H. Nasr and B. Bhanu, "Landmark Recognition for Autonomous Mobile Robots," *Proceedings of the 1988 IEEE International Conference on Robotics and Automation*, April 1988.

6. H. Nasr, B. Bhanu, and S. Schaffer, "Guiding an Autonomous Land Vehicle Using Knowledge-Based Landmark Recognition," *Proceedings of the DARPA Image Understanding Workshop*, February 1987, pp. 432-439.

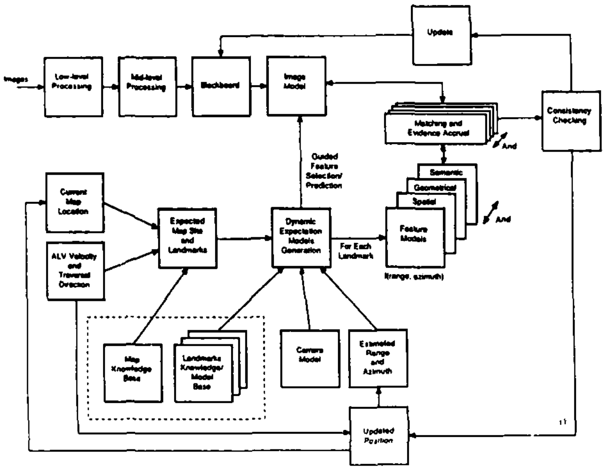


Figure 1. Detailed conceptual approach of PREACTE and DMM.

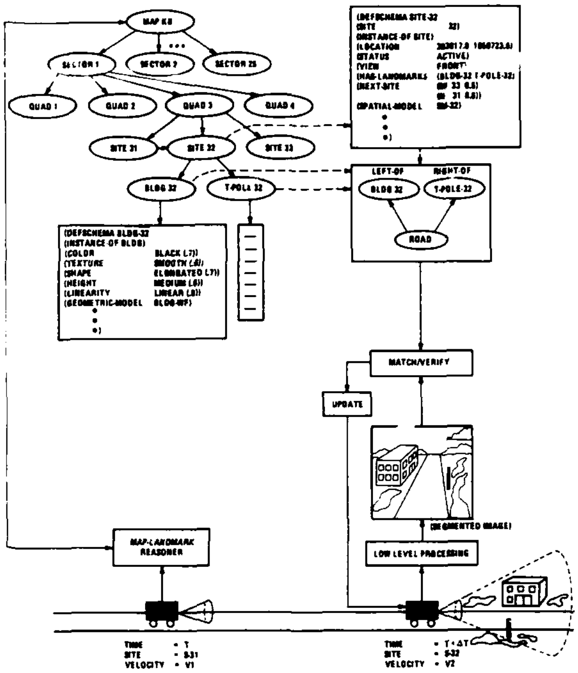


Figure 2. A graphic illustration of PREACTE's landmark recognition and map/landmark representation.

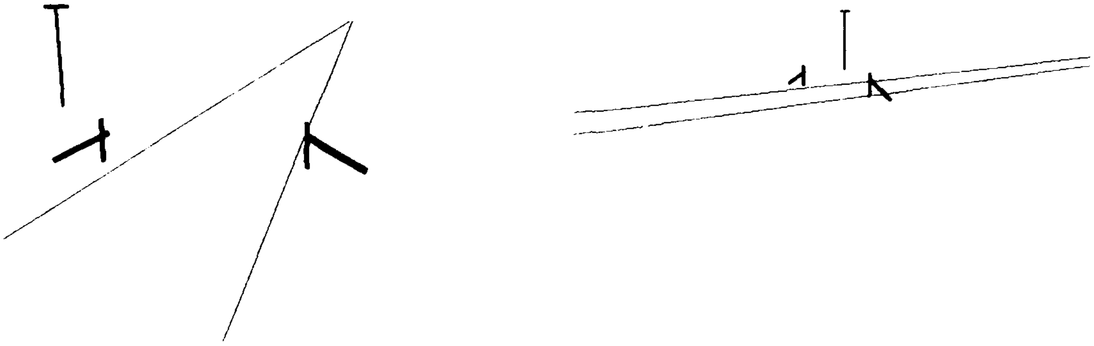


Figure 3. 2-D projections from different view angles and ranges.

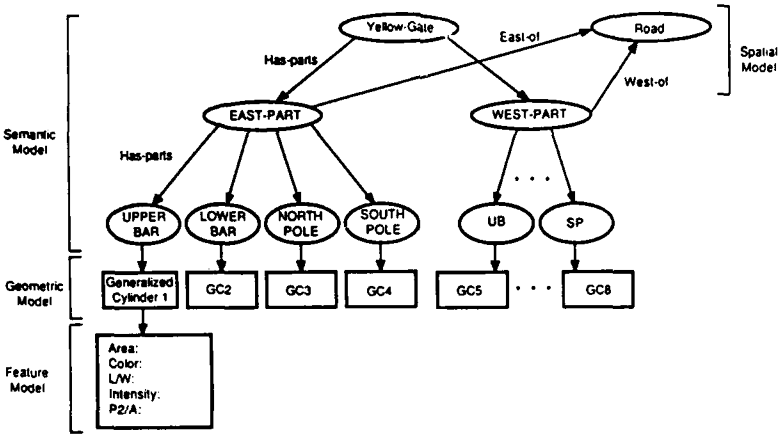


Figure 4. Hybrid model of the yellow gate landmark.

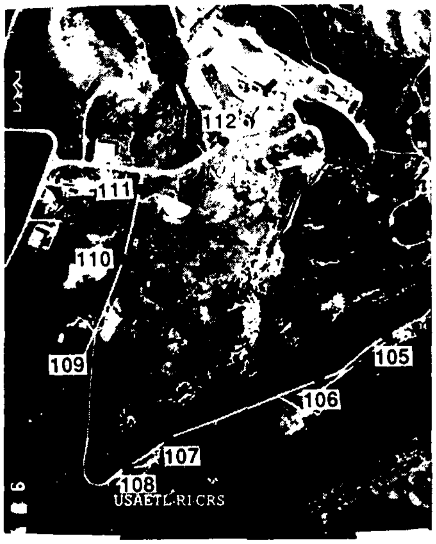
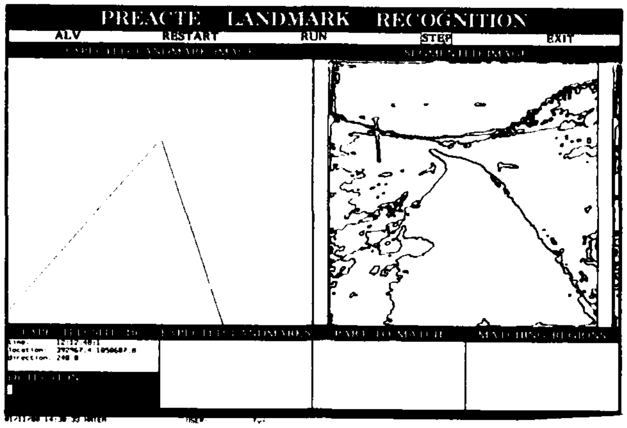


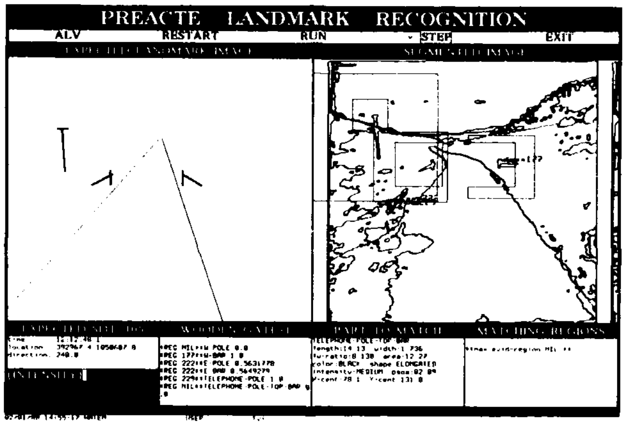
Figure 5. Aerial map photograph with selected sites for landmark recognition.



```
Time:       12:12:48.1
location:   392967.4 1050687.0
Direction:  240.0
```

DETECTION

Figure 6. Site 105 is the next predicted map site. It contains a gate and a te'ephone pole. PREACTE projects a road model of the scene and an image is processed and segmented.



```
Time:       12:12:48.1
location:   392967.4 1050687.0
direction:  240.0
```

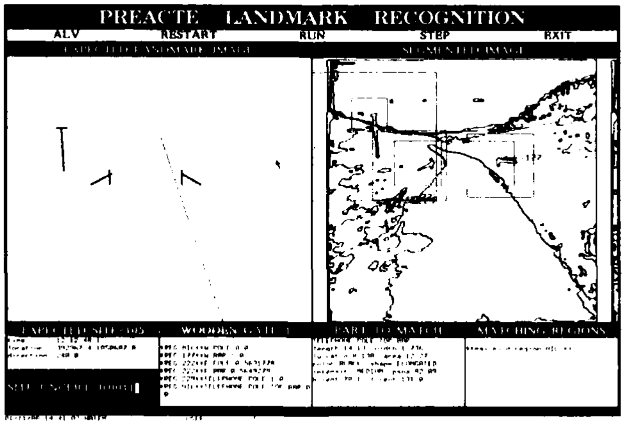
INTENSITY

```
#REG NIL**W-POLE 0.0
#REG 177**W-BAR 1.0
#REG 222**E-POLE 0.5631778
#REG 222**E-BAR 0.5649279
#REG 229**TELEPHONE-POLE 1.0
#REG NIL**TELEPHONE-POLE-TOP-BAR 0.0
```

```
TELEPHONE-POLE-TOP-BAR
length:14.13  width:1.736
l/w-ratio:8.138  area:12.27
color:BLACK  shape:ELONGATED
intensity:MEDIUM  ppos:82.89
X-cent:78.1  Y-cent:131.0
```

```
#max-avid-region NIL **
```

Figure 7. A 2-D model of the expected site is generated at the predicted range, and matching occurs. The "PART TO MATCH" pane shows descriptions of specific landmark parts as matching occurs. The intensity feature is emphasized in the lower left corner.



```
Time:       12:12:48.1
location:   392967.4 1050687.0
direction:  240.0
```

SITE UNCERTAINTY [illegible]

```
#REG NIL**W-POLE 0.0
#REG 177**W-BAR 1.0
#REG 222**E-POLE 0.5631778
#REG 222**E-BAR 0.5649279
#REG 229**TELEPHONE-POLE 1.0
#REG NIL**TELEPHONE-POLE-TOP-BAR 0.0
```

```
TELEPHONE-POLE-TOP-BAR
length:14.13  width:1.736
l/w-ratio:8.138  area:12.27
color:BLACK  shape:ELONGATED
intensity:MEDIUM  ppos:82.89
X-cent:78.1  Y-cent:131.0
```

```
#max-avid-region NIL **
```

Figure 8. End of detection stage, with site uncertainty computed.

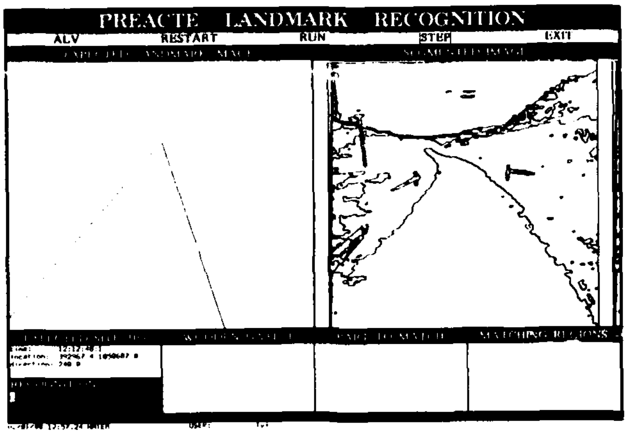


Figure 9. Beginning of recognition stage. New images are processed and a road model is projected.

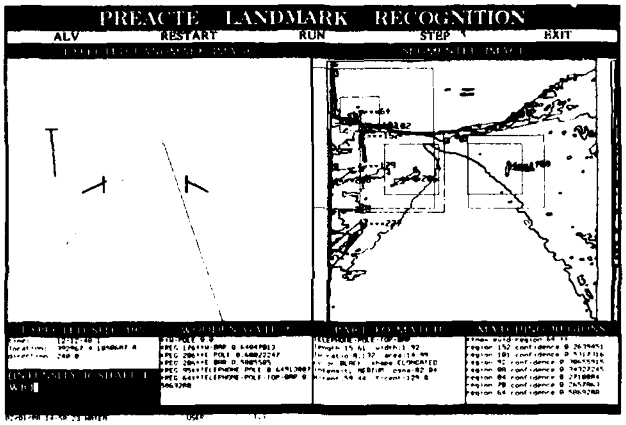


Figure 10. A new 2-D model of the scene is generated. More gate parts are identified. The rectangles over the segmented image indicate the FOAAs. The new model emphasizes a different set of features: intensity, length to width ratio, and a shape measure.

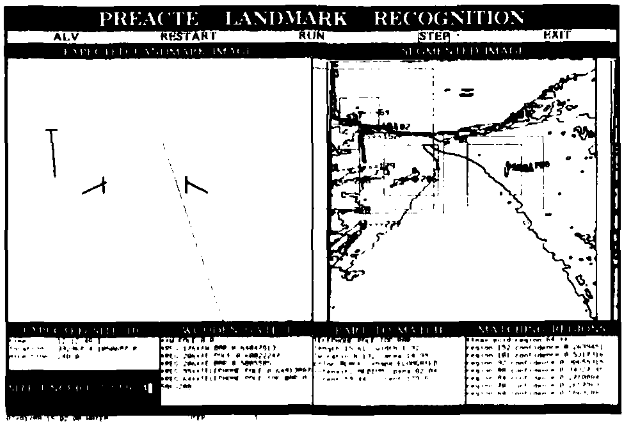
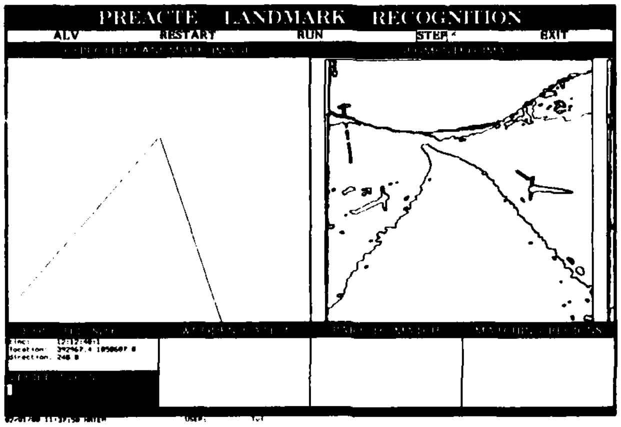
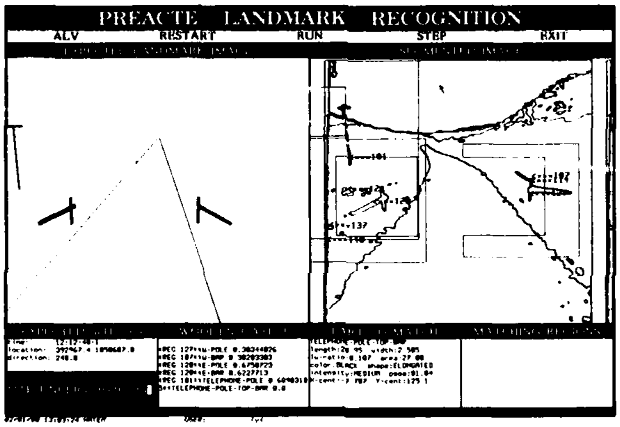


Figure 11. Site uncertainty has decreased because of additional positive evidences about the landmark.



```
time:       12:12:48:1
location:   392967.4 1050687.0
direction:  248.0
```

Figure 12. End of recognition stage and beginning of verification stage.

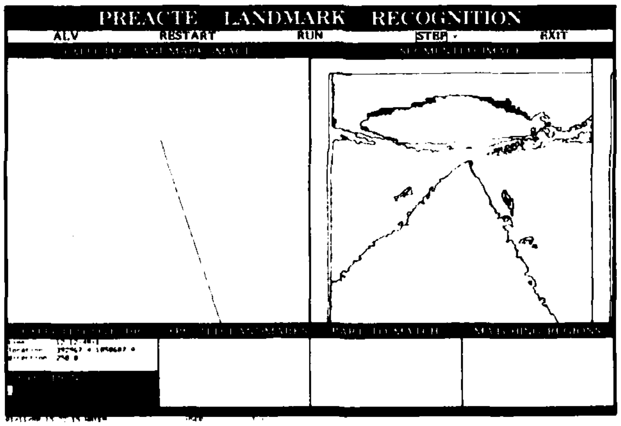


```
time:       12:12:48:1
location:   392967.4 1050687.0
direction:  248.0
```

```
#REC 127**U-POLE 0.30344026
#REC 107**U-BAR 0.30203303
#REC 120**E-POLE 0.6750723
#REC 120**E-BAR 0.6227713
#REC 101***TELEPHONE-POLE 0.6090310
3**TELEPHONE-POLE-TOP-BAR 0.0
```

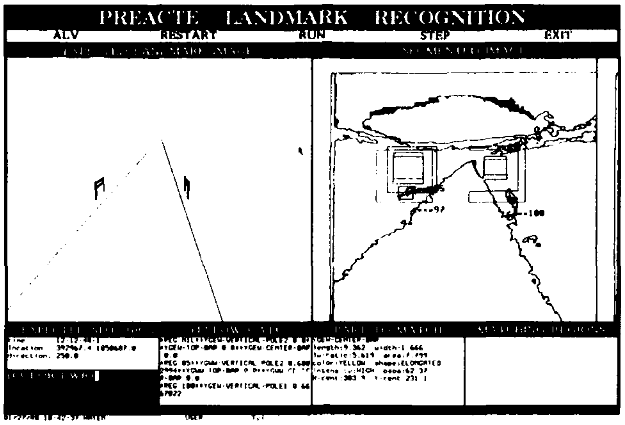
```
TELEPHONE-POLE-TOP-BAR
length:20.95  width:2.585
lw-ratio:0.107  area:27.00
color:BLACK  shape:ELONGATED
intensity:MEDIUM  pose:01.0*
X-cent:-2.787  Y-cent:125.1
```

Figure 13. Site uncertainty is computed at this verification stage. The uncertainty has slightly increased because of higher matching requirements.



```
time:       12:12:48:1
location:   392967.4 1050687.0
direction:  248.0
```

Figure 14. The vehicle arrives at a new site, which contains a yellow gate.

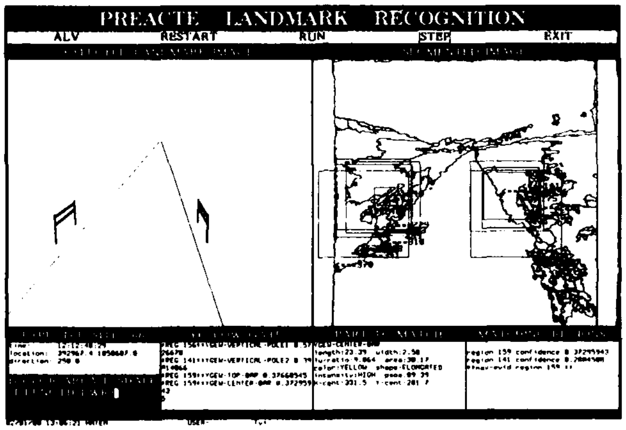


```
time        12:12:48:1
location    392967.4 1050687.0
direction.  250.0
```

```
#REG 101**YGEM-VERTICAL-POLE2 0 0
#YGEM-TOP-BAR 0 #**YGEM-CENTER-BAR
 0.0
#REG 85**YGEM-VERTICAL-POLE2 0.600
2994**YGEM-TOP-BAR 0 #**YGEM-CE
P-BAR 0.0
#REG 100**YGEM-VERTICAL-POLE1 0.66
67022
```

```
YGEM-CENTER-BAR
length:9.362  width:1.666
lw-ratio:5.619  area:7.799
color:YELLOW  shape:ELONGATED
intensity:HIGH  pos:62.37
x-cent:303.9  y-cent 231.1
```

Figure 15. Only the left part of the gate is detected. The right part of the gate falls mostly outside the FOAA.



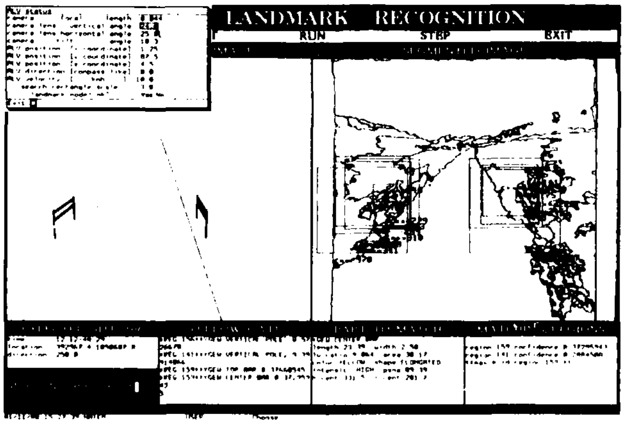
```
time:       12:12:48:29
location:   392967.4 1050687.0
direction:  250.0
```

```
#REG 156**YGEM-VERTICAL-POLE1 0.37
26670
#REG 141**YGEM-VERTICAL-POLE2 0.39
814064
#REG 159**YGEM-TOP-BAR 0.37660545
#REG 159**YGEM-CENTER-BAR 0.372959
43
5
```

```
YGEM-CENTER-BAR
length:23.39  width:2.58
lw-ratio:9.064  area:30.17
color:YELLOW  shape:ELONGATED
intensity:HIGH  pos:89.39
x-cent:331.5  y-cent:201.7
```

```
region 159 confidence 0.37299943
region 141 confidence 0.2884508
#max-evid region 159 ::
```

Figure 16. Five features are emphasized at this recognition stage: color, size, shape, length, and LWR.



```
ALV status
camera              focal          length
camera lens      vertical angle
camera lens horizontal angle    25
camera          tilt           angle
ALV position  [x coordinate]
ALV position  [y coordinate]   87.5
ALV position  [z coordinate]
ALV direction [compass line]   0.0
ALV velocity  [     kmh     ]  10.0
    search rectangle scale     1.0
      landmark model ok?       Yes No
```

```
time        12:12:48:29
location    392967.4 1050687.0
direction   250.0
```

```
#REG 159**YGEM-TOP-BAR 0.37660545
#REG 159**YGEM-CENTER-BAR 0.372959
```

```
length 23.39  width 2.58
intensity HIGH  pos 89.39
```

```
region 159 confidence 0.37299943
region 141 confidence 0.2884508
```

Figure 17. The box in the upper left corner shows the different camera parameters, which can be modified automatically or manually.

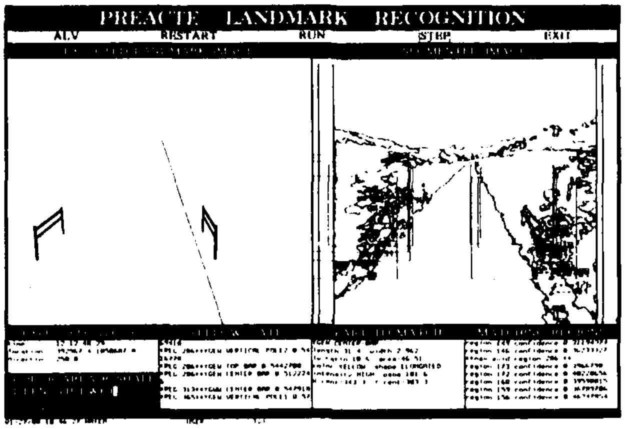


Figure 18. The gate at closer range, still at a recognition stage. Matching results have degraded because of bad segmentation results. The "Matching Regions" pane shows the candidate regions for a given part of the gate.

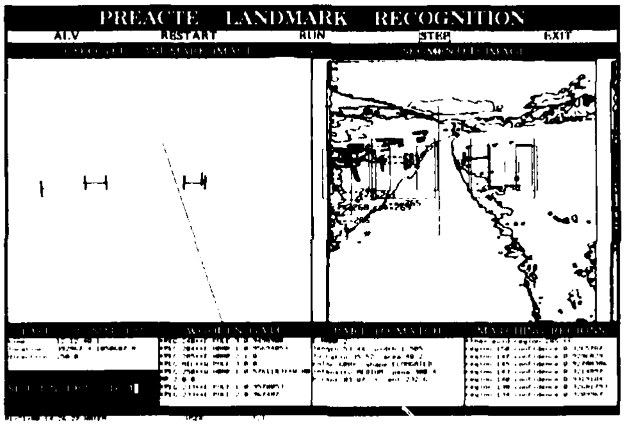


Figure 19. The vehicle arrives at another site that contains another gate.

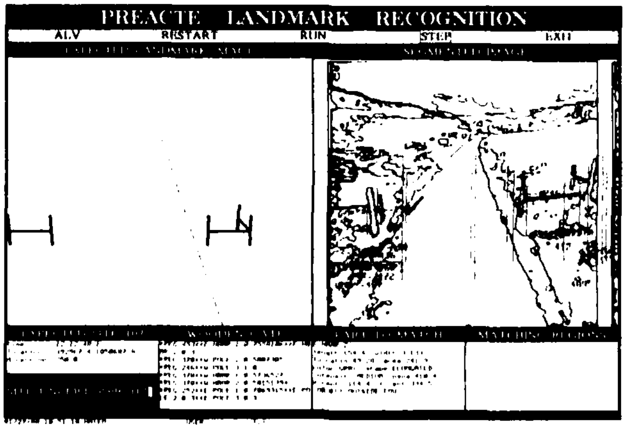


Figure 20. The gate model at very close range. The uncertainty value is still at an acceptable range. DMM predicts that some gate parts are outside the field of view and avoids matching them.

TRIPLE: A Multi-Strategy Machine Learning Approach to Target Recognition

Bir Bhanu and John C. Ming

Honeywell Systems and Research Center
3660 Technology Drive, Minneapolis, MN 55418

ABSTRACT

Current target recognition systems are unable to modify their behavior based on the dynamic environmental changes which occur around them. In order to perform robustly in unconstrained, outdoor environments, a target recognition system must be able to adapt its representation of this dynamic environment while maintaining acceptable performance levels. Machine learning technology offers some promising solutions to many of these problems faced in the target recognition scenario. Learning allows a system to use situation context, to adapt its representation of the changing environment, and to improve the system's recognition performance over time. This paper describes an innovative system which combines learning and target recognition into an integrated system which accomplishes the above tasks. This system is called **TRIPLE**: *Target Recognition Incorporating Positive Learning Expertise*. It uses two machine learning techniques known as explanation-based learning and conceptual clustering, combined with a knowledge-based reasoning system, to provide robust target recognition. A complete description of the TRIPLE system, as well as a simple example showing the system's behavior, is presented.

1. INTRODUCTION

Target recognition systems currently lack the ability to adapt to changing environmental conditions and to modify their behavior based on the context of the situation in which they are operating. In order to be effective in dynamic outdoor scenarios, a robust recognition system should be able to automatically acquire necessary contextual information from the environment. Most target recognition systems lack this capability. Their performance begins to quickly degrade when subjected to problems such as variable lighting conditions, image noise, and object occlusion.

Due to recent advances in machine learning technology, some of the problems encountered in the target recognition domain seem to be resolvable. Learning allows an intelligent recognition system to use situation context in order to understand images. This context, as defined in a machine learning scenario, consists of a collected body of background knowledge as well as environmental observations which may impact the processing of the scene. The vision-learning cycle of an advanced target recognition system would involve the following steps:

Sense - acquire an image and apply initial image processing algorithms.

Understand - using present background knowledge and scene observations, determine those objects of interest in the image.

Act - based on the objects found, act according to global system goals.

Update - modify system knowledge using image observations so performance will improve next time.

This cycle repeats each time the recognition system is required to produce results. Because the system dynamically reacts to the appropriate stimuli in the environment, it continuously adapts its internal knowledge to maintain acceptable performance levels.

In addition to the vision-learning cycle described above, other desirable features to be incorporated in an advanced target recognition system are: (a) the models used by the system to represent targets, contexts, and other system knowledge should be dynamic data structures. This specification allows the learning component to quickly modify the behavior of the system by changing the data on which it operates; (b) most data should be of a symbolic, qualitative nature, thus avoiding the problems encountered in dealing with quantitative information. Using qualitative information, we do not have to rely on obtaining precise geometric representations of target; (c) the system has to be able to handle problems such as imprecise segmentation, occlusion, noise, etc. Since advanced target recognition systems are required to operate in real world situations, they must be capable of handling these image problems that will inevitably occur; (d) the system should exhibit improved performance over time. This improvement may come in the form of faster recognition times, improved recognition accuracy, and higher confidence in system results.

Machine learning will facilitate two main breakthroughs in the target recognition domain: automatic knowledge base acquisition and continuous knowledge base refinement. The use of learning in the knowledge base construction will save the user from spending the enormous amount of time necessary to derive target models and databases. Knowledge base refinement can then be incorporated to make any necessary changes to improve the performance of the recognition system. These two modifications alone will serve to significantly advance the present abilities of most target recognition applications.

To validate the concept of a target recognition system with integrated machine learning capabilities, this paper presents an overview of a new approach to target recognition. The system currently under implementation is called

TRIPLE: *Target Recognition Incorporating Positive Learning Expertise.* The system uses a multi-strategy technique; two powerful learning methodologies are combined with a knowledge-based matching technique to provide robust target recognition. Using dynamic models, TRIPLE can recognize targets present in the database. If necessary, the models can be refined if errors are found in the representation. Additionally, TRIPLE can automatically store a new target model and recall it when that target is encountered again. Finally, since TRIPLE uses qualitative data structures to represent targets, it can overcome problems such as image noise and occlusion.

The two main learning components of the TRIPLE system are Explanation-Based Learning (EBL) and Structured Conceptual Clustering (SCC). Explanation-based learning provides the ability to build a generalized description of a target class using only one example of that class. Structured conceptual clustering allows the recognition system to classify a target based on simple, conceptual descriptions rather than using a predetermined, numerical measure of similarity. While neither method, used separately, would provide substantial benefits to a target recognition system, they can be combined to offer a consolidated technique which employs the best features of each method.

The remainder of this paper will provide the reader with most of the details of the TRIPLE system. To fully explain some of the machine learning techniques which are being utilized, section 2 presents a brief overview of machine learning, including the explanation-based learning and conceptual clustering methods. Following the coverage of these techniques, section 3 outlines the approach to target recognition used in TRIPLE. Section 4 provides an example of TRIPLE's abilities using automobiles as the targets to be recognized. Finally, section 5 presents the conclusions of the research to date and discusses future plans for the current system.

2. MACHINE LEARNING TECHNIQUES

The ability to reason and the ability to learn are the two major capabilities associated with an intelligent being. Today, machines have been given the ability to reason by developing algorithms which duplicate, to a limited extent, a reasoning process. Unfortunately, machines do not have the ability to modify or update this reasoning process if the situation so dictates. Nor do they have the ability to learn new concepts which are encountered in the processing of information. The lack of both of these abilities has severely limited the effectiveness of computers to work in unconstrained environments.

Fortunately, this situation is slowly beginning to change. Progress is being made in the development of algorithms which can modify their behavior as the environment in which they operate changes. Machine learning now exists as a valid portion of the AI and psychology fields, has dedicated conferences and journals, and is being added as a feature in a broad variety of computer science applications. While many of these additions are very superficial and do little to improve the performance of the system to which they are added, research has led to fundamental advances in several areas of machine learning technology.

Advances in the machine learning field originated in the late 50's. Rosenblatt[12] developed the perceptron, a self-organizing system, although it never achieved the success anticipated by many of its advocates. Following this effort, researchers concentrated on tasks such as learning from examples and language acquisition which relied on significant amounts of data and time to search the problem space. Also known as inductive learning systems, these learning from example techniques were applied in a wide range of application domains. The most influential research performed in the area of inductive learning was accomplished by Winston.[16] His structural learning system formed concept descriptions from a set of carefully selected examples of the concept as well as "near misses". Near misses represent concepts that are similar to the one being learned, differing only in a small number of very significant details. Positive examples serve to generalize the concept while near misses provide the necessary amount of specificity. Many inductive learning systems followed the early work done by Winston. Dietterich and Michalski[5] present a good comparison of various systems which incorporate learning from examples.

In most systems which utilize inductive learning, a method known as *generalization* is used to extract the common features which characterize a group of objects. Generalization has been used in various AI contexts for many years, although the results have been difficult to compare due to substantial differences in implementation and domain of application. Mitchell[10] casts the generalization problem into a search framework and compares various approaches to the problem. The search space consists of the possible generalizations that can be constructed for a given problem. Methods of generalization can then be characterized by a search strategy such as depth-first, breadth-first, or version space technique.

Connell and Brady[2] developed a system which learns the descriptions of two-dimensional objects including aerial views of airplanes or silhouette images of various hand tools. This technique produced structured production rules which were used to recognize subsequent instances of similar objects. Using inductive generalization techniques which allow for disjunctions, Connell and Brady's method was one of the first systems which could learn from real image data.

The trend in machine learning has been to incorporate techniques which can derive the maximum amount of information from single examples, using analytic methods rather than empirical ones. Current research is now directed at developing programs which provide learning from observation and discovery. Explanation-base learning (EBL) and structured conceptual clustering (SCC), both of which are used in the TRIPLE system, are examples of learning methodologies which employ a high level of inference. EBL, classified as a *learning by observation* technique, uses inference to construct a useful concept description from a single example of that concept. SCC, which is also a *learning from observation* method, employs an even higher level of inference since it does not rely at all on any user input to classify a group of targets into conceptually simple groups. These techniques will now be discussed.

2.1 Explanation-Based Learning

Most of the early systems which utilized learning from examples were able to achieve impressive results compared to methods which did not use any form of machine learning at all. However, it was discovered that the user may find it difficult or impossible to provide the learning mechanism with enough examples to properly generalize the concept description. Additionally, the system was unable to justify the generalization which was produced from a set of examples; the user could not obtain a description of how the objects had been generalized. These factors led to the development of learning systems which, using applicable background knowledge, could generate a concept description from a single, user-provided example. At the same time, the system also created an explanation as to why the example yielded that particular generalization. Initially referred to as *explanation-based generalization* (EBG), this technique is now commonly called *explanation-based learning* (EBL).

The generalization process employed by EBL can be viewed as a search through the possible concept description space. The objective is to locate the correct definition of the concept being learned. To constrain the size of this search space, EBL relies on knowledge of the p⁻oblem domain. Since the extra information present in a set of multiple examples is not provided, EBL must use some other type ᶜ knowledge to sufficient!y generalize the single example. This information exists in the form of relevant background knowledge which is given to the system. Using the background knowledge, EBL is able to produce a valid generalization of the single example. Additionally, it creates a justification of the generalization in terms of the background knowledge used to produce that generalization. This justification is called the *explanation* of the concept example.

The origin of the explanation-based approach to machine learning can be traced back to the STRIPS system developed by Fikes *et al.*[6] which learns generalized robot path planning motions. From this initial work, the creation of a concept description from a single example was then formalized by DeJong.[3] In this paper, he introduces the term *explanation-based generalization.* As DeJong continued working on his system, others began work on their own extensions or changes to the initial EBG method. The most prominent of these was the research done by Mitchell, Keller, and Kedar-Cabelli.[11] They proposed a standardized approach to explanation-based generalization. This technique creates an explanation structure, represented as a proof tree, which serves as the generalization of the concept. The generalization is a two step process. The first step forms the explanation that separates the relevant and irrelevant feature values present in the training example. Second, the explanation is analyzed to determine the constraints (numeric values, numeric ranges, or enumerated values) on the feature values which will allow the explanation to apply in general.

In response to perceived inadequacies in the work by Mitchell *et al.*, DeJong and Mooney[4] proposed further revisions to the EBG system. DeJong and Mooney felt that the term *explanation-based learning* was more complete than explanation-based generalization since the approach seemed to be applicable to both concept refinement and concept generalization. This version of EBL serves as the basis for the target model creation and refinement component of the TRIPLE system described in section 3. While Mitchell *et al.*'s version of EBG produced the object generalization in a two step process, the EBL technique simultaneously forms an explanation of the training example and builds the generalized concept of the training example. In addition, EBL is capable of specializing a previously-defined, over-generalized object concept. This refinement ability is very valuable since it provides a partial solution to the problem of generalizing non-independent conjunctive sub-goals. In other words, after several passes over a concept description which may contain conflicting information, EBL is capable of properly representing this concept, while EBG would have failed. Since this effectively provides a form of explanation-based specialization as well as explanation-based generalization, DeJong and Mooney have aptly named the method explanation-based learning.

2.2 Structured Conceptual Clustering

Classification of similar objects has traditionally been accomplished using mathematical techniques such as numerical taxonomy and cluster analysis.[1, 14, 15] Using a pre-defined set of object features or attributes, these techniques would compute clusters of objects; clusters are characterized by high intra-class similarity and low inter-class similarity. Clustering methods are generally unable to identify groups of objects which represent conceptually simple concepts since they rely on numerical measures of similarity. In addition, the results usuall‹ must be interpreted by expert data analysts to decipher the classification results.

Problems with numerical clustering techniques and the proposed solutions have been numerous. However, they do not address some of the fundamental problems inherent with the clustering methods. First, numerical clustering techniques are context free. They make no use of any contextual or background information while computing object similarities. Psychological tests have shown that humans make use of significant amounts of context when classifying objects. Second, most methods are unable to expand the feature space in order to discover new features which may yield ideal classifications. Simple, linear combinations of object attributes can often be used to locate intrinsic groups of data. Third, the techniques do not have the ability to select and evaluate object attributes when generating clusters. The attributes are simply used to compute distances between neighboring objects and clusters. Finally, the classification results still have to be interpreted because a characterization of the clusters is not produced.

The problems mentioned above have caused researchers to design systems which try to model the classification techniques used by humans. People normally group objects using a conjunction of attributes which represent conceptually simple ideas. At the same time, they also consider the context in which the objects act, which often determines important features which can be used in the classification task. Using a collection of background knowledge to provide context, Michalski[7] developed a new version of clustering which identified groups of objects which represent the same type of conceptually simple ideas that humans tend to use. This approach to classification is known as *conceptual clustering*. Since it does not rely on a teacher to preclassify the objects, conceptual clustering is superior to earlier systems which use learning from examples.

An implementation of the technique Michalski developed is presented in a paper by Michalski and Stepp.[9] This method, known as the CLUSTER/2 program, constructs a classification of objects only if a given class can be specified by a conjunctive concept which uses selected object attributes. Quality measures such as the fit between the clustering and the observed events, the inter-cluster distance, total number of features used in the concept description, and the number of features which individually discriminate among all the clusters have been used to judge the quality of the selected object attributes.

To validate the performance of the CLUSTER/2 algorithm and compare its performance with classical clustering approaches, Michalski and Stepp[8] tested the classification ability with 18 other numerical taxonomy methods. Only 4 of the 18 numerical methods were able to produce the conceptually appealing results of the CLUSTER/2 program. These results show that conceptual clustering achieves many of the classification goals used by humans.

As a further extension to their work, Stepp and Michalski[13] constructed a new conceptual clustering system which incorporates three main changes from the previous technique: objects are complex and require structural descriptions; relevant attributes may not be initially provided and should be dynamically determined in that case; and rules of inference are used to derive useful high-level concepts from the initial low-level information. This new version of conceptual clustering is called CLUSTER/S. To produce valid classifications, the system is provided with a general goal of classification. Using the supplied goal, the system then references the collection of background knowledge to determine the relevant attributes and features useful for clustering.

The background knowledge is organized into a network structure called a Goal Dependency Network (GDN). The information present in the GDN can represent general-purpose knowledge as well as domain-specific knowledge, both of which are necessary in the problem solving process. General-purpose knowledge is made up of fundamental constraints and criteria which specify the general properties of classification. The domain-specific information contains inference rules for deriving new descriptors and rules for determining which descriptors will be relevant. Given a high level goal of classification, the GDN specifies the related sub-goals and any associated object attributes which are relevant at that level in the network. If the relevant descriptors are not present, the background knowledge is used to derive new descriptors which are applicable. The Goal Dependency Network plays a vital role in the construction of meaningful classifications.

The ability of CLUSTER/S to handle complex, structural objects relies on the information present in the GDN. Using the goal of classification, the system is able to determine which structural elements of the object are most useful in selecting the appropriate classification scheme. The advantages of this approach include:

(1) Ability to handle compound objects which require structural descriptions in order to be easily classified.

(2) Use of goal-directed inference from information provided by the GDN.

(3) Formulation of new object attributes using the GDN.

(4) Allows the system to be model or data driven.

The ability to select appropriate classification features and to generate new attributes when necessary places the conceptual clustering technique into the area of machine learning referred to earlier as *learning from observation.*

In the domain of target recognition, most complex targets are represented as a structured collection of sub-parts. The ability to cluster such descriptions is very useful in this application. In the remainder of this paper, Stepp and Michalski's CLUSTER/S system will be referred to as Structured Conceptual Clustering (SCC) because of the ability to handle these structural descriptions.

3. TRIPLE TARGET RECOGNITION SYSTEM

Although machine learning has been used in many applications, very little effort has been made to combine several learning techniques together. Learning methodologies are used independently to provide adaptive ability and improved system performance. However, TRIPLE incorporates explanation-based learning and conceptual clustering into a multi-strategy learning approach to target recognition. By utilizing the capabilities of each learning method at appropriate steps in the recognition and learning process, TRIPLE overcomes the inherent limitations present in these learning techniques. EBL's main limitation is the matching time required when the number of target models becomes large. SCC's has problems with model biases when the number of object class examples is small. Combining the ability of EBL to characterize a target using a single training example with SCC's efficient method of organizing objects once they have been properly modeled yields an integrated learning system which handles the target recognition task.

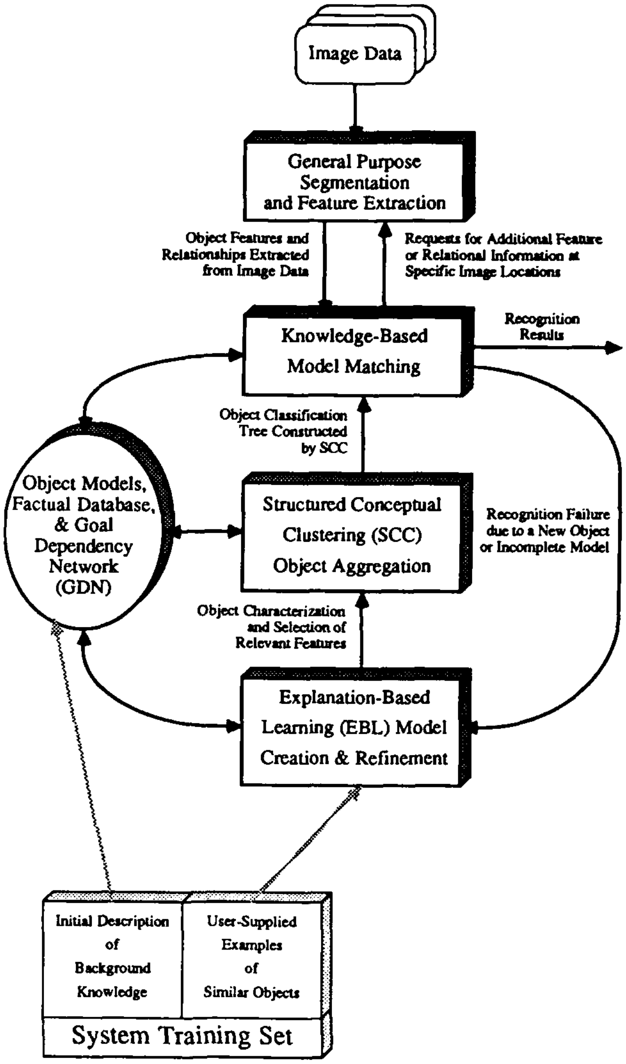


Image Data
General Purpose Segmentation and Feature Extraction
Object Features and Relationships Extracted from Image Data
Requests for Additional Feature or Relational Information at Specific Image Locations
Knowledge-Based Model Matching
Recognition Results
Object Classification Tree Constructed by SCC
Object Models, Factual Database, & Goal Dependency Network (GDN)
Structured Conceptual Clustering (SCC) Object Aggregation
Recognition Failure due to a New Object or Incomplete Model
Object Characterization and Selection of Relevant Features
Explanation-Based Learning (EBL) Model Creation & Refinement
Initial Description of Background Knowledge
User-Supplied Examples of Similar Objects
System Training Set

Figure 1: The TRIPLE target recognition system

Figure 1 shows the various components of the TRIPLE target recognition system. Basically, the characterization abilities of explanation-based learning are combined with the aggregation capabilities of the conceptual clustering technique. The EBL component is used to create and refine target models while the SCC component is used to structure the EBL-generated models into an efficient classification tree. The domain rules and facts relevant to the recognition problem are stored in the background knowledge database. The goal and sub-goal information is stored in a modified version of a Goal Dependency Network (GDN) originally used by conceptual clustering. The GDN has been adapted so that the EBL component can access the necessary information.

The TRIPLE system, as indicated in Figure 1, is composed of six main components:

(1) A system training set which initializes the target recognition process.

(2) A target database, factual knowledge, and a goal dependency network. These items are combined to form the background knowledge pertaining to the selected recognition domain.

(3) An explanation-based learning component which provides generalized target descriptions.

(4) A structured conceptual clustering system which arranges the current set of target descriptions into a classification tree.

(5) A model matching component which uses the existing classification tree to recognize an unknown target description.

(6) A segmentation and feature extraction component which processes the images and provides the necessary object features and relationships.

Each of these elements will be described in detail in the following sub-sections, followed by a description of the recognition-learning cycle used by the TRIPLE system.

3.1 System Training Set

The initial input to the TRIPLE system consists of a set of target examples and a collection of background knowledge pertinent to those targets. For each target which will be recognized by TRIPLE, the user must supply a set of 1 or more representative examples for that target class. Each set of examples will be analyzed by the EBL system and a generalized target description will be created. Note that the input data for each target class need only consist of a single example since EBL is capable of producing a correct generalization from only one target. However, as will be discussed in section 3.3, EBL will be able to generalize a single conceptual description if provided with more than one example.

In addition to the examples for each target which will be recognized, the user must also provide the initial description of the goal dependency network and any factual knowledge which may be necessary when the learning systems attempt to make inferences on the target models. The GDN contains the goal hierarchy necessary for the recognition and learning systems to derive the applicable target attributes when recognizing new targets, refining existing models, or processing incomplete data.

3.2 Target Models, Factual Database, and Goal Dependency Network

The collection of background knowledge used by the TRIPLE system is composed of three main groups of data: target models, factual data, and a goal dependency network. The target models, which are originally built by the EBL component, are stored separately in the target model database in case they are needed for refinement purposes later. Although the target model information will also be represented in the classification tree, the actual schema which defines each model is kept in the database for use by the EBL system when characterizing a target class. The features and relationships which are labeled as relevant by the EBL component are tagged in the schema and will be used in the SCC system. The models are dynamically modified by the recognition and learning elements when missing or incorrect attributes and relationships are discovered.

The factual knowledge originally given by the user is maintained in the fact database. This knowledge consists of a set of rules and facts which define the domain in which the recognition process is operating. For example, if the recognition domain is automobiles, the factual knowledge may contain information regarding plausible locations, orientations, uses, and other contextual data for different type of vehicles on the road. This information is used in conjunction with the learned target models by the EBL, SCC, and recognition components. As the various components attempt to apply this information, inconsistencies or gaps may be found. These problems will propagate the assertion of new facts and the retraction of incorrect or useless facts. Thus, the factual database will be as dynamic as the models which depend on this data.

The modified version of the GDN in the TRIPLE system is primarily used by the EBL component when determining attribute relevancy. Although it was originally designed to be used by the SCC system, the GDN information is used in the EBL phase of the recognition loop. The version of the GDN used in the EBL phase contains two high level goals: creation of a new target model and refinement of an existing target model. The selection of the appropriate goal is described in the next section. Using the goal which is received, the GDN accesses the goal-subgoal hierarchy and selects the attributes and relationships which are useful in characterizing the particular target. This alternative application of the GDN information is possible since the main responsibility of the GDN is to provide the SCC component with a set of relevant target features; this task is now accomplished by the EBL system.

3.3 Explanation-Based Learning

The explanation-based learning component of the TRIPLE system is responsible for two critical functions:

(a) Processing the training examples provided by the user.

(b) Explaining target recognition failures to improve system performance.

Each of these tasks will now be described in detail.

(a) Processing training examples:

The first step in the target recognition process is to acquire and represent a set of target models. This job has traditionally been very difficult due to the amount of work necessary to generate a correct model. The TRIPLE system uses the power of EBL to simplify the modeling process. Since EBL can generalize a target description from a single example, the user merely provides a set of target attributes and relationships in the training set. From this data, EBL selects the relevant attributes and relationships using the modified version of the GDN. Other applicable background knowledge which provides useful data transformations is used here as well. These transformations are used to infer high-level target attributes from the various input attributes.

The schema which is used to store this new target will still retain the entire list of attributes and relationships which were originally provided to the EBL system. The relevant attributes which have been selected will be tagged in order to separate them from the rest of the attributes. The remaining attributes are retained in case model refinement is ever needed. Some attributes will be indicated in the GDN as statistical attributes that the recognition component uses to provide another adaptive capability to the recognition system. If any of the attributes are statistical, they will be initialized at this time. This process will be more fully described in section 3.5.

Since all training examples are processed sequentially before the target recognition process begins, the training phase also insures that the target generalizations are not identical. This situation can result from a collection of background knowledge which is not diverse enough to distinguish the various training examples. Alternatively, the set of target attributes selected by the user may not be broad enough to individually characterize each target class. In either case, the EBL system performs a simple comparison against all existing schemata during the training phase. If an identical match occurs, the user will be notified by the TRIPLE system and must then supply the additional data in order to separate those target classes. After the training phase, this problem will not occur again because the unknown targets are first checked against all existing models before the learning loop is ever entered. Thus, it is impossible for EBL to create the same generalization twice, except during this stage.

An additional extension to the standard EBL technique is the capability to generate a single explanation from more than one representative example. This ability is useful when the user is trying to characterize a target model that may not be sufficiently captured in a single training example, such as when modeling a 3D target. Using only a single example in this situation could bias the resulting generalization towards that particular example. In order to prevent the possibility of this bias, the EBL component of the TRIPLE system allows the user to create a single target model with up to 3 examples of that target. There are two ways to accomplish this goal. First, each of the examples can be individually generalized and these generalizations can be intersected to provide a single model. Alternatively, the similarities of each example can be located and used to provide a generalization. The first method is preferable since it is easier to intersect the generalizations due to the smaller number of remaining attributes and relationships. Multiple examples will hopefully eliminate any biases which may be implied from using only a single example.

Using the above modifications to the standard EBL technique, the target model training task has been effectively automated in the TRIPLE system. After characterizing each of the targets, a copy of the schema will be stored in the background knowledge database and another copy will be sent to the SCC component for integration into the classification tree.

(b) Explaining recognition failures:

The main task of the EBL component is explaining the various types of recognition failures that occur. EBL must be provided with two pieces of information in order to properly explain these failures. First, the system must have the target description which caused the recognition failure. Second, the type of failure which resulted from this target must also be determined. The failure type is used as the goal concept used in the generalization process. Most EBL systems require that the user provide the goal concept from which the generalization can be derived. However, the TRIPLE system does not require the user to provide this kind of information. Instead, the model matching component determines the appropriate goal concept and sends it to the EBL system. The goal concepts are generated when the recognition system is not able to properly recognize a new target. TRIPLE's EBL component is designed to process two main types of recognition failures:

(1) Failures caused by the presence of a new target.

(2) Failures due to an incomplete or incorrect target model.

Failures which result from encountering a new model are handled in the same manner as when processing the user-supplied training examples. This process was described previously in this section.

If an incorrect model causes the recognition failure, EBL can explain this event and alter the model to avoid the problem in the future. There are several types of target model errors which EBL can effectively handle. If the current model contains an attribute or relationship which is not relevant, EBL will remove the relevant attribute or relationship tag from that feature in the model schema. If the model contains an attribute whose value is not useful, EBL can modify the value. Finally, if a target model is not using an attribute or relationship which is relevant, it can be tagged as relevant in the model schema. Since these types of recognition failures are located and labeled by the model matching component, EBL can minimize the search space when trying to locate the incorrect data.

3.4 Structured Conceptual Clustering

Normally, conceptual clustering applications classify hundreds of examples simultaneously, many of which belong to the same class, thus requiring the use of a similarity-based method to correctly characterize the targets present in a given cluster. The similarity-based method performs the characterization task defined in Section 4. However, in the TRIPLE system, SCC receives only one pre-characterized example which represents a target class from the EBL system. The EBL component has already performed the task usually handled by the GDN; all relevant target attributes and relationships have been identified. Since SCC relies on conceptual simplicity as a quality measure, instead of numeric quality values which depend on the number of samples in a cluster, it can produce a valid clustering which contains only one sample per cluster.

The measure of simplicity determines how precisely the classification tree distinguishes various targets. For example, assume the target recognition domain is automobiles. If the measure of simplicity is very coarse, it may be impossible for the system to separate different types of cars, although cars and trucks are distinguishable. As the measure of simplicity becomes more complex, different types of cars can be identified (sedan, coupe, etc.).

To provide greater efficiency in the conceptual clustering process, SCC does not always create a new classification tree. If a model refinement operation has been selected, only a portion of the classification tree has to be changed. By tracing the classification tree until the branch at which the incorrect feature is located, SCC can merely re-cluster the rest of that branch. Since target models will be isolated along only one branch of the tree, there is no danger in rearranging individual branches of the tree.

If a new model is added to the classification tree, the tree will have to be reconstructed since new attributes may be present in the new model which are not present in the tree. In addition, the new model may interact with the existing target models in such a way that features which were not previously useful as important classifiers can now be used to distinguish target classes. Thus, the conceptual clustering process will be applied to the entire group of target models, including the new target which has been created. The classification tree can then be passed to the model matching component for use in identifying unknown targets.

3.5 Knowledge-Based Model Matching

The model matching component of the TRIPLE system uses the classification tree produced by SCC in order to identify unknown targets. The model matching component has access to the knowledge base so that any necessary data transformations or inferences can be made. This component has two functions within the TRIPLE system: recognition of targets which already exist in the model database and identification of the two types of recognition failures which cause the system to enter the learning loop.

When an unknown target is received by the recognition system, it is first checked against all items currently in the model database. This matching procedure is done by using the classification tree. Since the model schemata are represented in a search tree format, the recognition process is far more efficient than comparing the new target with each model individually. Because some of the features present in a target model may be missing in the image observations due to segmentation problems or occlusion, the model matching component monitors the parsing of the classification tree to deciue on the type of recognition or failure which has occurred. Each of these possibilities will now be briefly discussed.

Complete Matching: If the model matching component can successfully traverse the classification tree, match all necessary attribute and relationship criteria, and arrive at a leaf node which contains a target model, that model has been correctly matched. During the parsing of this tree, the recognition element has some flexibility in terms of slight variations in attribute values and a small number of missing attributes. However, if any attributes are missing, they must be minor in nature; global target features can not be safely ignored. Confidence in a successful match is computed based on the amount of variation in attribute values as well as the significance of any missing attributes.

Incomplete Matching: If the model matching component can correctly parse the high level nodes of the classification tree (these nodes specify global attributes of the target models) prior to exhausting all available image data, all targets present in any leaf nodes at the end of that branch can be hypothesized as valid models for the data which is present. Matching global features and relationships implies that the target is lacking the details necessary for a precise identification. However, this information can guide the segmentation and feature extraction algorithm which can be prompted to provide additional information on the target being identified. If no more feature information can be obtained, the model matching component provides the complete set of possible target identifications.

Occlusion: If the recognition process is not able to match any high level classification attributes, but can recognize many low level attributes and relationships, the presence of occlusion is very probable. In this case, a confidence level can be computed based on the reliability of the features which lead to the matching. For example, if the global features of a car (length, height, etc.) cannot be matched, but the shapes of the hood, bumper, headlights, and fenders are present, the recognition system can use this information to identify the target as a car. A confidence threshold is used to insure that the model matching component does not match ridiculously simple features and report a valid matching.

Model Refinement: Model refinement is determined by a process which is very similar to complete recognition. The parsing of the classification tree should proceed without missing more than a few target attributes or relationships. However, the failure is encountered when comparing the values stored in the attributes themselves. While the attribute itself may be present in the target model, the values in the model attributes may be different than the values in the image attributes. This situation implies that the target model contains most of the correct attributes but needs to be refined by updating the values present in the attribute variables. In addition, any missing or extra attributes present in the target model can be inspected during the refinement process to determine whether they should be included or removed, respectively.

New Target Model: This situation is encountered when the model matching component can not apply any meaningful portions of the classification tree to the data received from the image. The image data is passed to the learning loop of the TRIPLE system for characterization and integration into the classification tree. If the data contains enough high level and low level details to properly characterize a new target model, it will be processed by the EBL system and incorporated into the classification tree. Otherwise, the EBL system will report the failure to understand the data and the object will be classified as unknown.

The recognition element updates the statistical variables for the given target model if and when that model is used to recognize a target. This process allows the system to gradually determine that certain features have a higher utility since they are always used in the recognition process. In addition, the variable values of certain attributes can be slowly modified by the recognition process in order to overcome any initial bias that may have been derived from the initial construction of the target model. This factor is especially useful for those models which are added by the learning system during normal operations. While the user is responsible for providing examples which are highly representative of a target class during the training phase, target models acquired during the recognition phase of the system may be subject to a much higher level of noise. Thus, the model matching component can slowly adapt these models as more targets of this type are recognized and the system determines a better value for many of the attributes which characterize that target. Changes in attribute values made by the model matching component will be very gradual compared with the changes which result from activating the model refinement process during the learning process of the TRIPLE system.

3.6 Segmentation and Feature Extraction

The general purpose segmentation and feature extraction component of the TRIPLE system extracts the necessary low level features and relationships from the image data which are then used by the knowledge-based model matching component in the target recognition process. The feature extraction process is designed to locate the most prominent features in the image data and to determine the significant relationships between these features. The features of interest in the image include regions, edges, corners, arcs, and ellipses.

Once the primitive features have been located, the feature extraction process then considers the spatial relationships between various features. If a predefined orientation of a set of features is located, the feature extraction process hypothesizes the presence of a high level symbolic feature. For example, in the domain of automobile recognition, if a pair of concentric ellipses is found in the image, the feature extraction component will hypothesize the presence of a tire and wheel. This symbolic feature can then be used in conjunction with other symbolic features in the model matching process described in the previous section.

In addition to providing initial feature and relational information, which is a data driven process, the feature extraction component can also be used in a model driven manner by the model matching component. During the parsing of the classification tree, the model matching procedure may encounter the presence of a symbolic feature in the tree which is not present in the initial symbolic feature list provided by the feature extraction component. The model matching process can request the feature extraction procedure to reanalyze a specific portion of the image in search of that particular feature. The parameter sets and thresholds used in this instance can be more relaxed than during the initial image processing since only a portion of the image is being processed and the presence of a specific feature is being sought. If after several attempts to find the feature, the extraction process is not successful, the feature extraction component will abandon the search and the model matching process will be informed of the failure. Otherwise, the presence and location of the desired feature will be returned to the model matching component.

3.7 Recognition-Learning Process

The vision-learning cycle of the TRIPLE system consists of three main steps: model creation/refinement, target aggregation, and target recognition. A brief review of each of these stages as well as how they interact in the target recognition process in provided in the rest of this section.

The user provides an initial collection of background knowledge and a set of training examples which represent the targets needed for recognition purposes. The domain background knowledge is stored for use by all components of the TRIPLE system. The sets of target examples are given to the EBL system for characterization. During the training phase, the EBL component must insure that the target models which are created are unique. If they fail to meet this requirement, the user should be notified and must then provide additional background knowledge or more detailed target examples. The model schemata are stored in the knowledge base for subsequent use. In addition, a copy of each schema is then sent to SCC for inclusion in the new system classification tree. Once SCC has received all the models from the training set, it creates the initial classification tree. Finally, the tree is sent to the model matching component for recognition of unknown targets.

After the initial training has been completed, the recognition component uses the classification tree to recognize each target processed by the TRIPLE system. If a target has already been modeled, it will be recognized, the confidence will be computed, and this information will be output to the user. The types of recognition produced by the TRIPLE system include complete recognition, incomplete recognition, and recognition in occluded scenes.

If the model exists, but needs to be refined, the recognition component will send the image data to the EBL system which will identify the incorrect data, update the model, and send this information to the SCC system. SCC will then make the necessary modifications to the classification tree and return the updated tree back to the recognition element, where this process begins again.

If the model does not exist, the image data is sent to the EBL system for construction of a new model. This procedure is subject to verification of the usefulness of the observed image data. Once the EBL system has built the new model schema, it is passed to the SCC component. In this case, the entire classification tree is rebuilt, in case the new model contains attributes which may lead to a more efficient tree representation. As before, the tree is then sent to the model matching component for use in subsequent identification activities.

4. EXAMPLE - VEHICLE RECOGNITION

In order to fully illustrate the interactions among the components of the TRIPLE system, this section provides a simple example using military vehicles as objects. The vehicles which have been chosen are examples of Soviet vehicles which would likely be encountered by an autonomous vehicle during a reconnaissance or surveillance mission. This group includes armoured personnel carriers, reconnaissance vehicles, and cargo transport vehicles. Complex objects such as tanks and self-propelled guns have been avoided to simplify this example. In addition, the vehicles which have been selected will be treated as 2D objects in terms of the features which are used to classify them. However, in practice, the TRIPLE system is being implemented using a 3D object modeling system which incorporates salient object features and relationships between those features.

Figure 2 shows the system training set for the selected Soviet vehicles used in this example. The set includes an armoured personnel carrier (BTR-70), a reconnaissance vehicle (BRDM-1), and a cargo truck (MAZ-502). The feature set defined for this example are simple features such as object length, height, wheelbase, number of wheels, and so on. Each of these features can be reliably extracted from an image containing that target. Note that not all features are defined on all objects; the cargo truck does not have an armament feature but does include a load height feature which specifies the height of the cargo platform.

The EBL component of the TRIPLE system takes the above training set and selects the relevant target features using the background knowledge for this particular target domain. In this example, the rules may contain information such as:

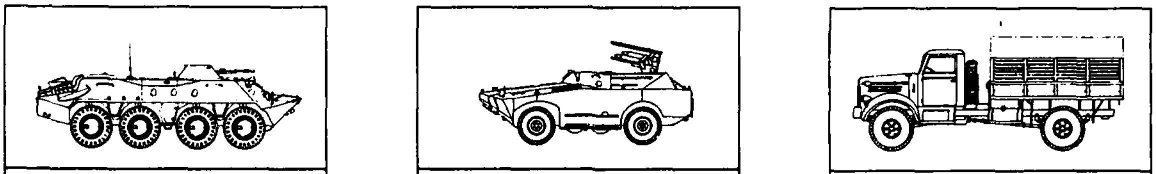
Vehicle length, height, and wheelbase are
always used in characterizing a target.

If a vehicle has armaments, they are useful
in characterizing that target.

The set of relevant features for the Soviet vehicles is shown in Figure 3. Since the armoured personnel carrier and the reconnaissance vehicle have armaments, that feature has been included in the target models in addition to length, height, and wheelbase. As mentioned in section 3, the remaining target attributes are retained in the target model database in case the system needs to use them in the future.

Once the EBL characterization for each target has been obtained, the SCC component creates the classification tree which will be used by the knowledge-based matching component during the subsequent recognition procedure. SCC segregates the targets based on the conceptual simplicity of the target classes. The classification tree for this example is indicated in Figure 4. Because the cargo truck contains no armaments, it is conceptually different from the other two vehicles and is placed in a separate branch of the classification tree. In the right branch of the tree, the length feature of those targets is determined to be the most useful in distinguishing between the recon vehicle and the personnel carrier. The resulting classification tree is sent to the matching component for use in recognizing these three targets.

The matching component continues to use the classification tree until it encounters a target which cannot



BTR-70 Personnel Carrier	
Length	7.54 m
Width	2.8 m
Height	2.24 m
Wheelbase	4.4 m
Track	2.38 m
Ground Clearance	.48 m
Tire Size	1.2 m
Number of Wheels	8
Armament	Machine Gun

BRDM-1 Recon Vehicle	
Length	5.7 m
Width	2.25 m
Height	1.9 m
Wheelbase	2.8 m
Track	1.6 m
Ground Clearance	.32 m
Tire Size	1.0 m
Number of Wheels	4
Armament	Missiles

MAZ-502 Cargo Truck	
Length	7.15 m
Width	2.7 m
Cab Height	3.03 m
Wheelbase	4.52 m
Track	2.03 m
Ground Clearance	.35 m
Load Height	1.48 m
Tire Size	1.2 m
Number of Wheels	4

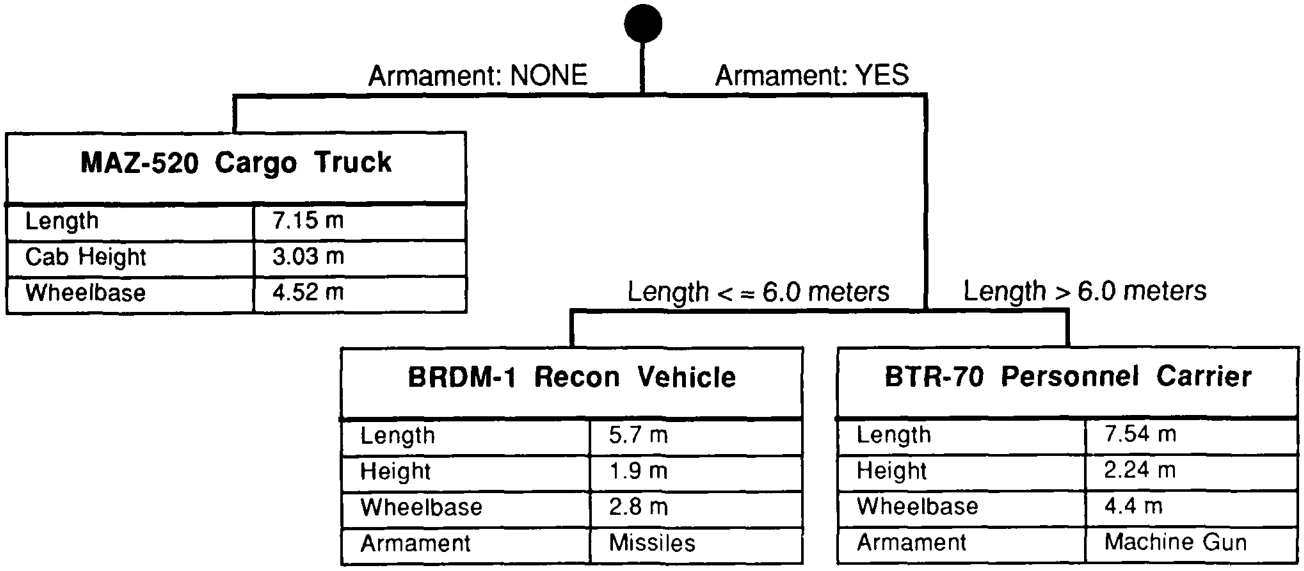
Figure 2: System training set for Soviet vehicles

BTR-70 Personnel Carrier	
Length	7.54 m
Height	2.24 m
Wheelbase	4.4 m
Armament	Machine Gun

BRDM-1 Recon Vehicle	
Length	5.7 m
Height	1.9 m
Wheelbase	2.8 m
Armament	Missiles

MAZ-502 Cargo Truck	
Length	7.15 m
Cab Height	3.03 m
Wheelbase	4.52 m

Figure 3: EBL-selected relevant target features



MAZ-520 Cargo Truck	
Length	7.15 m
Cab Height	3.03 m
Wheelbase	4.52 m

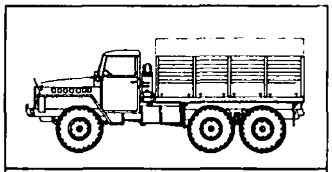
BRDM-1 Recon Vehicle	
Length	5.7 m
Height	1.9 m
Wheelbase	2.8 m
Armament	Missiles

BTR-70 Personnel Carrier	
Length	7.54 m
Height	2.24 m
Wheelbase	4.4 m
Armament	Machine Gun

Figure 4: SCC-generated target classification tree

be processed by the tree information. This situation was fully described in section 3.5. When this failure occurs, the unknown target is sent to the EBL component for complete characterization. This event signals the beginning of the learning cycle. Figure 5 shows a target which cannot be processed by the classification tree due to differences in the each of the three target features. The vehicle is assumed to be a new target and the feature list is sent to the EBL system. Figure 6 displays the results of the relevant feature selection by the EBL component. In the process of selecting relevant features, EBL has determined that the number of wheels is a useful feature when distinguishing the new vehicle from the MAZ-520 cargo truck which is also present. This feature is then marked in the MAZ-520 feature list as relevant. The resulting classification tree produced by the SCC component is shown in Figure 7. The number-of-wheels feature is now used in the left branch of the tree to distinguish between the MAZ-520 cargo truck and the new unknown vehicle.

This simple example has shown how the TRIPLE system can achieve target model acquisition and model refinement by explaining feature significance and using those features to efficiently recognize future instances of a target.

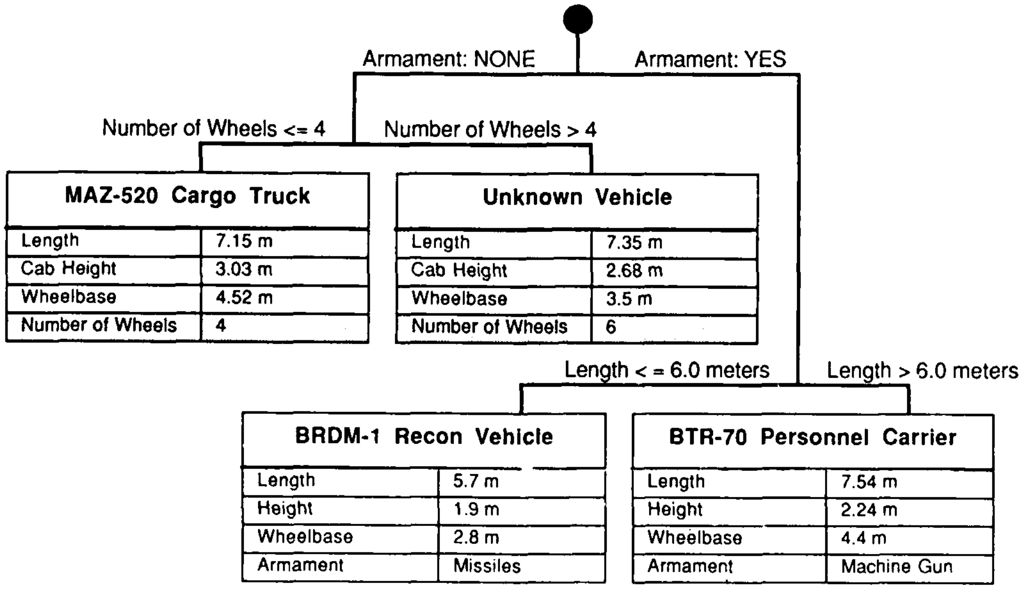


Unknown Vehicle	
Length	7.35 m
Width	2.69 m
Cab Height	2.68 m
Wheelbase	3.5 m
Track	2.0 m
Ground Clearance	.41 m
Load Height	1.42 m
Tire Size	1.2 m
Number of Wheels	6

Figure 5: Unknown vehicle which triggers the learning cycle

Unknown Vehicle	
Length	7.35 m
Cab Height	2.68 m
Wheelbase	3.5 m
Number of Wheels	6

Figure 6: EBL-selected relevant target features



MAZ-520 Cargo Truck	
Length	7.15 m
Cab Height	3.03 m
Wheelbase	4.52 m
Number of Wheels	4

Unknown Vehicle	
Length	7.35 m
Cab Height	2.68 m
Wheelbase	3.5 m
Number of Wheels	6

BRDM-1 Recon Vehicle	
Length	5.7 m
Height	1.9 m
Wheelbase	2.8 m
Armament	Missiles

BTR-70 Personnel Carrier	
Length	7.54 m
Height	2.24 m
Wheelbase	4.4 m
Armament	Machine Gun

Figure 7: New SCC-generated target classification tree

5. CONCLUSIONS AND FUTURE WORK

We have shown the usefulness of integrating machine learning with target recognition to create a system which adapts its representation of the target domain in order to operate effectively in an unconstrained, outdoor environment. Present target recognition systems do not have this capability.[17] Using the characterization ability of explanation-based learning and the efficient classification techniques of conceptual clustering, the TRIPLE system provides automatic knowledge-base acquisition and refinement. The system is robust in several respects: it can acquire a target model using a single example of that target; it uses only the most relevant features during subsequent recognition stages; and target model refinements are facilitated through the use of the EBL component.

Presently, we are working on an implementation which recognizes complex, 3D targets such as cars, trucks, vans, etc. This system uses 3D object models which consist of object features and relationships among the features. In addition, we will evaluate the performance of the TRIPLE system when subjected to problems such as occlusion, image noise, etc.

REFERENCES

1. B. Bhanu and J.C. Ming, "Recognition of Occluded Objects: A Cluster-Structure Algorithm," *Pattern Recognition* **20**(2) pp. 199-211 (1987).

2. J.H. Connell and M. Brady, "Generating and Generalizing Models of Visual Objects," *Artificial Intelligence* **31**(2) pp. 159-183 (1987).

3. G. DeJong, "Generalizations Based on Explanations," Proc. of 7th International Joint Conference on Artificial Intelligence, pp. 67-69 (1981).

4. G. DeJong and R. Mooney, "Explanation-Based Learning: An Alternative View," *Machine Learning* **1**(2) pp. 145-176 (1986).

5. T.G. Dietterich and R.S. Michalski, "A Comparative Review of Selected Methods for Learning from Examples," pp. 41-81 in *Machine Learning: An Artificial Intelligence Approach*, ed. R.S. Michalski, J.G. Carbonell, & T.M. Mitchell,Tioga Publishing Co., Palo Alto, California (1983).

6. R.E. Fikes, P.E. Hart, and N.J. Nilsson, "Learning and Executing Generalized Robot Plans," *Artificial Intelligence* **3**(2) pp. 251-288 (1972).

7. R.S. Michalski, "Knowledge Acquisition through Conceptual Clustering: A Theoretical Framework and Algorithm for Partitioning Data into Conjunctive Concepts," *Intl Journal of Policy Analysis and Information Systems* **4** pp. 219-243 (1980).

8. R.S. Michalski and R.E. Stepp, "Automated Construction of Classifications: Conceptual Clustering Versus Numerical Taxonomy," *IEEE Transactions on Pattern Analysis and Machine Intelligence* **5**(4) pp. 396-409 (1983).

9. R.S. Michalski and R.E. Stepp, "Learning from Observation: Conceptual Clustering," pp. 331-363 in *Machine Learning: An Artificial Intelligence Approach*, ed. R.S. Michalski, J.G. Carbonell, & T.M. Mitchell,Tioga Publishing Co., Palo Alto, California (1983).

10. T.M. Mitchell, "Generalization as Search," *Artificial Intelligence* **18**(2) pp. 203-226 (1982).

11. T.M. Mitchell, R.M. Keller, and S.T. Kedar-Cabelli, "Explanation-Based Generalization: A Unifying View," *Machine Learning* **1**(1) pp. 47-80 (1986).

12. F. Rosenblatt, "The Perceptron: A Perceiving and Recognizing Automaton," Cornell Aeronautical Laboratory Report No. 85-460-1 (1957).

13. R.E. Stepp and R.S. Michalski, "Conceptual Clustering of Structured Objects: A Goal-Oriented Approach," *Artificial Intelligence* **28**(1) pp. 43-69 (1986).

14. G. Stockman and J.C. Esteva, "Use of Geometrical Constraints and Clustering to Determine 3-D Object Pose," Proc. of 7th International Conference on Pattern Recognition, pp. 742-744 (1984).

15. G. Stockman, S. Kopstein, and S. Benett, "Matching Images to Models for Registration and Object Detection via Clustering," *IEEE Transactions on Pattern Analysis and Machine Intelligence* **PAMI-4**(2) pp. 229-241 (1982).

16. P.H. Winston, "Learning Structural Descriptions from Examples," in *The Psychology of Computer Vision*, ed. P. Winston,McGraw-Hill, New York (1975).

17. B. Bhanu, "Automatic Target Recognition: State of the Art Survey," *IEEE Transactions on Aerospace & Electronic Systems* **AES-22**(4) pp. 364-379 (1986).

USING FLOW FIELD DIVERGENCE FOR OBSTACLE AVOIDANCE IN VISUAL NAVIGATION

Randal C. Nelson
John (Yiannis) Aloimonos

Computer Vision Laboratory
Center for Automation Research
University of Maryland
College Park, MD 20742-3411

ABSTRACT

The practical recovery of quantitative structural information about the world from visual data has proven to be a very difficult task. In particular, the recovery of motion information which is sufficiently accurate to allow practical application of theoretical shape from motion results has so far been infeasible. Yet a large body of evidence suggests that use of motion is an extremely important process in biological vision systems. It has been suggested by Thompson [Thom86] that qualitative visual measurements can provide powerful perceptual cues, and that practical operations can be performed on the basis of such clues without the need for a quantitative reconstruction of the world. The use of such information is termed "inexact vision". This paper describes the investigation of one such approach to the analysis of visual motion. Specifically, the use of certain measures of flow field divergence were investigated as a qualitative cue for obstacle avoidance during visual navigation. It is shown that a quantity termed the *directional divergence* of the 2-D motion field can be used as a reliable indicator of the presence of obstacles in the visual field of an observer undergoing generalized rotational and translational motion. Moreover, the necessary measurements can be robustly obtained from real image sequences. A simple differential procedure for robustly extracting divergence information from image sequences which can be performed using a highly parallel, connectionist architecture is described. The procedure is based on the twin principles of directional separation of optical flow components and temporal accumulation of information. Experimental results are presented showing that the system responds as expected to divergence in real world image sequences, and the use of the system to navigate between obstacles is demonstrated.

I. INTRODUCTION

Navigation is a basic operation in automation which must be performed by any system that manipulates or responds to physical objects within its environment. A robot assembly arm, for instance, must move without accidentally striking the workpiece or any other obstacle, and at a higher level, must be able to locate and acquire tools or subassemblies. In existing industrial systems, such navigation is typically performed by following a previously determined path either in absolute, or more flexibly, in relative coordinates. This is more or less the computer equivalent of a cam. Such methods have the advantage of being efficient with regard to the use of computational resources, and are relatively straightforward to program and analyze. On the negative side, preprogrammed solutions are inflexible with respect to variations in the task, and intolerant of changes in the workspace. A computer-driven welder following a preset path does not do a good job if the workpiece is half an inch from the anticipated position. There is thus considerable incentive for developing systems capable of performing active navigation by using sensory feedback to direct movement.

Visual navigation has received particular attention in this respect, primarily because of its potential, as manifested in biological organisms, to provide a large amount of information about many different characteristics of the environment. Animals use vision to obtain information about location, shape, movement, and identity of objects. Designers of automatic systems would like to be able to design artificial systems with similar capabilities; however, doing so has generally proved to be extremely difficult.

One reason for this difficulty is that extracting visual information from images is simply a lot of work. The visual cortexes of animals that perform complex visually moderated behaviors contain millions of neurons, each of which performs computations which by the lowest estimates require thousands of computer steps per second to simulate, and possibly much more. Much of this capacity is probably necessary to carry out whatever cortical image processing occurs.

We feel, however, that another reason for the limited success in many efforts is that the fundamental approach is, in a certain sense, overambitious. There is a natural tendency to attempt to reduce multiple visual tasks to some lowest common denominator. Thus, for example, the problem of visual navigation has often been considered a subproblem of the "shape from x" problem, and is mentioned as an application of the more general studies designed to extract accurate, quantitative information about the three dimensional structure of the world. However, despite numerous elegant theoretical results, none of the shape from x studies have yielded systems which can actually be used to navigate in a real environment. The usual situation is that a mathematical solution to the general problem can be obtained only under specific assumptions which are not particularly valid in the real word. On the other hand, many practical applications have been so concerned with performing one narrowly defined operation as to render them devoid of useful general principles.

The support of the Defense Advanced Research Projects Agency and the Center for Night Vision and Electro-Optics is gratefully acknowledged

There is, however, another approach. Rather than obtaining a specific solution to a general problem, e.g., solving the shape from flow problem assuming piecewise smoothness and exact information, one may seek a general solution to a specific problem, e.g., solving the visual obstacle avoidance problem with no specific restrictions on the shape of the environment. If this approach is taken, then qualitative measurements which are specific to the problem at hand, but which can be obtained practically in real-world environments, can be used. This is the approach taken in this paper.

The above strategy recalls the program proposed by Rodney Brooks [Broo86] for achieving artificial intelligence through building robots. The claim is made that understanding of intelligent behavior can be approached by attempting to construct working mechanisms which duplicate progressively more sophisticated abilities displayed by living animals. Thus, for example, before one is able to navigate, one must be able to wander without hitting things. Before one is able to wander, one must be able to walk. Before one is able to walk, one must be able to stand, and so forth. A computational theory of some ability should invoke no "lower level" process for which understanding has not been demonstrated by an ability to build a mechanism which actually performs the process. Any theory which does not conform to this requirement is on very shaky ground at best. The basic premise here is that to be believable, a theory of machine vision must be demonstrated by constructing a mechanism which implements it. Given the many plausible sounding theories of vision which have turned out to be either unimplementable or so incomplete as to be unusable in a practical sense, there are good grounds for maintaining some such position.

One of the most elementary forms of navigation is obstacle avoidance by a moving, compact sensor. It is a prerequisite, however, for many more complex abilities since any system performing a more complicated task must avoid obstacles in the process. Obstacle avoidance is thus one specific problem for which a general solution is highly desirable. In this context, a general solution refers to a system that works effectively in a wide range of real environments. This implies, among other things, that the system performance does not depend upon artificial constraints on the nature of objects in the environment such as assuming planar or smoothly curved surfaces, rigid or unmoving objects, mathematically uniform textures, and so forth. For reasons to be discussed, we believe that qualitative motion measurements are particularly appropriate for this task.

The use of motion information in natural image processing systems is both widespread and fundamental. The visual systems of many simple organisms are essentially blind to fixated images. Even in the human visual system, if the saccadic movement of the eyes is suppressed or canceled, the perceived image fades from view. In what might be considered a hierarchy of natural vision systems, from light-sensitive spots responsible for phototropic behavior in some single celled organisms to those of the higher vertebrates, the simplest that display capabilities which would be useful in the design of automated systems appear at the level of the flying insects, which perform a wide variety of complex, visually moderated behaviors, including flight navigation in three dimensional environments, pursuit and capture of prey (e.g., dragonflies), and location and identification of food sources (e.g., honeybees). Available evidence suggests that the insect visual system responds primarily to moving images, and that such information is critical in orientation and navigation [Reic76]. These systems, the simplest capable of visual navigation and obstacle avoidance, appear to rely almost completely on motion derived information. There is thus a biological motivation for investigating the role of movement in visual navigation.

In this paper, we explore the use of a particular motion cue, namely, flow field divergence, which corresponds to the intuitive notion of image expansion. This is a relatively simple cue which can be derived locally and in parallel from fine-grained time sliced image sequences, i.e, those in which the motion from one frame to the next is generally on the order of, or less than, one pixel. Motion information is accumulated over time from local differential relations, without establishing any correspondence, and without use of regularization procedures. It turns out that the flow information thus obtained is sufficiently accurate to be useful in navigation.

Our approach, and the organization of this paper, reflect all three levels in the paradigm introduced by Marr [Marr82], namely, computational theory, algorithms, and implementation. Sections II and III present the theoretical basis for our use of divergence measurements. Section IV describes the design of a practical obstacle avoidance system at the algorithmic level. Section V describes the implementation of the system, and the results of experiments run using the implementation.

II. INTRODUCTION TO MOTION ANALYSIS

A camera moving within a three dimensional environment produces a time-varying image which can be characterized at any time t by a two dimensional vector-valued function f known as the *motion field*. The motion field describes the two dimensional projection of the three dimensional motion of scene points relative to the camera. Mathematically, the motion field is defined as follows. For any point (x,y) in the image, there corresponds at time t a three dimensional scene point (x',y',z') whose projection it is. At time $t+\Delta t$, the world point (x',y',z') projects to the image point $(x+\Delta x,y+\Delta y)$. The flow field at (x,y) at time t is given by

$$f(x,y,t) = \lim_{\Delta t \to 0}\left[\frac{\Delta x}{\Delta t},\frac{\Delta y}{\Delta t}\right].$$

Figure 1 shows motion fields for several situations.

The motion field depends on the motion of the camera, the three dimensional structure of the environment, and the three dimensional motion (if any) of objects in the environment. If all these components are known, then it is relatively straightforward to calculate the motion field. From a computer vision standpoint, the interesting question.is whether the process can be inverted to obtain information about camera motion and structure of the environment. This is not so easy, and if arbitrary shapes and motions are permitted in the environment, there may not be a unique solution. However, it can be mathematically demonstrated that, in most situations, a unique solution exists.

The existence of such solutions has inspired a large body of work detailing the mathematical theory of extracting shape and/or motion information from the motion field. Some approaches utilize flow information at separated locations under the assumption of three dimensional rigidity of the environment [Ullm79]. Others use the flow and its derivatives in a local neighborhood under some assumption about the structure of environmental surfaces (e.g., they are planar). Several authors have recently obtained solutions to the general shape from flow problem in closed form with a set of linearized equations [Tsai84 Long81]. The end result of such analytic approaches is a set of equations relating the flow field and/or its spatial derivatives to the camera motion and the three-dimensional structure of the environment. Most of these studies, however, have started with the assumption that detailed and accurate motion-field information is available. Unfortunately, the solutions to the equations are frequently inordinately sensitive to small errors in the motion field. Tsai and Huang [Tsai84] report 60% error for a 1% perturbation in input for some instances using their method. This error sensitivity is due both to inherent ambiguities in the flows produced by certain camera motions, at least over restricted fields of view, and to the reliance on differentiation of the flow field which amplifies the effect of any error present in the data. Because the extraction of accurate flow fields from real image sequences has proven to be extremely difficult [Ullm81], such analytically based shape from motion schemes have so far found little practical application. Recent theoretical research has addressed the development of noise insensitive solution procedures (e.g., [Yasu85]) and methods for smoothing inconsistent flow information extracted from image sequences (e.g., [Anan85]).

Several methods have been proposed for extracting motion information from image sequences. These methods fall into two rough categories: those based on time and space derivatives of the images, pioneered by Horn and Schunk [Horn81], and matching techniques similar to those employed in stereo vision [Barn80].

The correspondence methods attempt to identify corresponding features in consecutive image frames, and compute the motion field from the positional change. This approach has two problems; first, the correspondence problem is itself quite difficult since features may change from one image to the next, and even appear and disappear completely. Second, since each correspondence gives the flow at just one point, it may prove impossible to find enough strong features to densely determine the flow.

Differential methods, on the other hand, attempt to determine the flow field from local computations of the spatial and temporal derivatives of the gray scale image. The problem with this approach is that the local apparent motion of the image, known as the *optical flow*, does not necessarily correspond to the 2-D motion field. The most obvious demonstrations of this fact are pathological examples. For instance, a spinning, featureless sphere under constant illumination has zero optical flow, but a non-zero motion field. Conversely, a stationary sphere under changing illumination has non-zero optical flow, but zero motion field. Verri and Poggio [Verr87] have shown that these are not just pathological examples due to the lack of features, but that only under special conditions of lighting and movement do the motion field and the optical flow correspond exactly. They also show, however, that for sufficiently high gradient magnitude, the agreement can be made arbitrarily close. This corresponds to the intuition that for strongly textured images the motion field and the optical flow are approximately equal.

A more serious problem with differential techniques is that only the component of the optical flow parallel to the local image gradient can be recovered from local differential information. This is commonly referred to as the *aperture problem*. Intuitively, the aperture problem corresponds to the fact that for a moving edge, only the component of motion perpendicular to the edge can be determined. This fact is responsible for the illusion of upward motion produced by the rotating spirals of a barber pole where either vertical or horizontal motion could produce the local motion of the edges, and the eye chooses the wrong one. In order to determine both components of the flow field vector, information must be combined over regions large enough to encompass significant variations in the gradient direction.

In brief, the correspondence methods typically yield full information at scattered points, while the differential methods yield partial information over a relatively dense set of points. Despite a great deal of effort expended in devising flow invariants, regularization methods, and matching techniques, neither approach has yielded data sufficiently accurate to allow the theoretical results to be reliably applied. Adiv [Adiv85] argues that inherent near ambiguities in the 3-D structure from motion problem may make the goal of extracting information sufficiently precise to allow uniform application of the theoretical solutions unattainable in practice. Verri and Poggio [Verr87] make essentially the same point, arguing that the disagreement between the motion field and the optical flow makes the computation of sufficiently accurate quantitative values impractical.

Despite the above problems, a few studies have actually made use of motion information extracted from real-world image sequences. Lawton [Lawt83] robustly obtained the direction of sensor translation and the relative depth of a set of feature points from image sequences taken by a camera restricted to translational motion, using an adaptation of the generalized Hough transform. The technique worked, however, only when the sensor motion was constrained to two degrees of freedom, as the parameter space was otherwise of unmanageable size. More recently, Lucas [Luca85] reported a method of obtaining all five motion parameters, given a reasonable initial guess, using a method based on gradient descent in the parameter space to match highly smoothed images. In order to achieve this, images were averaged over neighborhoods of up to 60 by 60 pixels, which allows the motion parameters to be determined within limits, but yields little or no detailed flow field information which could be used to analyze the environment. Both these experiments achieved their success by limiting their scope to a particular practically motivated aspect of the general problem.

III. FLOW FIELD DIVERGENCE AS A QUALITATIVE MOTION CUE

Quantitative measurements of the flow field appear to be difficult to make, yet motion seems to be extremely important in even the most primitive natural vision systems. It is possible of course, that natural vision systems are simply much better at extracting quantitative motion data than current artificial systems. Another possibility, however, is that such systems make use of more qualitative properties of the flow field which can be calculated more robustly. It is this second possibility which interests us.

Consider the problem of visual obstacle avoidance by a compact, moving sensor. The solution of the general shape from motion problem would certainly allow a safe course to be navigated, but the converse is not true, i.e., possession of knowledge about a safe course does not necessarily permit the construction of an accurate depth map. Thus solving the problem of visual obstacle avoidance does not presuppose the solution of the general shape from motion problem. If the only requirement is for the system to avoid collisions, substantially less interpretation of the visual information is needed, which considerably simplifies the problem.

For the purposes of obstacle avoidance, it is sufficient to detect those objects having relative motion towards the sensor in time to alter course and avoid a collision. Exact information about the shape and velocity of the obstacle is not necessary. A variety of qualitative cues, which are computationally simpler to robustly evaluate than accurate global flow fields, can indicate the presence of such obstacles. For instance, the approach of a large, visually textured object produces an expanding region in the image which would thus indicate a danger zone. For small obstacles, movement relative to local background motion is a similar cue. In neither case is accurate quantitative information necessary in order to avert collision. Such cues have the further advantage that, being fairly local, they do not depend for their existence on global assumptions such as the three dimensional rigidity of the environment. An expansion detector would warn of an approaching object whether it was moving independently or not.

A practical system can be envisioned which would evaluate a number of such movement cues, and combine them to dynamically maintain a retinal "hazard map" which could be used to navigate between obstacles. In such a system, it is not necessary that the cues be absolutely specific. A certain number of false positives (for instance, a net divergence detector could be triggered by rotating objects under certain circumstances) can be tolerated, so long as they are not frequent enough to overly clutter the hazard map. False negatives can also be tolerated as long as another subsystem detects the danger. In general, some errors are probably inevitable in systems based on qualitative measurements; they may occasionally fail. The evidence that the best biological systems can be fooled by appropriate "optical illusions" lends support to this view. Because of the complexity of the real world, it may also be unfeasible to mathematically characterize the conditions under which such a system is guaranteed to work. However, we believe that it is feasible to design systems which will satisfactorily perform practical tasks without recourse to a quantitative reconstruction of the world.

A qualitative measurement which is particularly useful in the context of obstacle avoidance during visual navigation is the divergence of the motion field. This is one of the potential candidates mentioned by Thompson in his position paper on "inexact" vision [Thom86] The obvious motivation stems from the fact that an obstacle in relative motion towards the camera produces an expanding image, i.e., one whose image flow has positive divergence. Thus regions of positive divergence represent potential obstacles with the distance to the obstacle positively correlated to the magnitude of the divergence. It can also be shown that divergence is invariant under rotational motion of the sensor which is a valuable property because it allows the cue to be utilized even when the sensor is not completely stabilized.

Approaching obstacles do not, however, represent the only situation producing positive divergence in the motion field. A tilted surface translating parallel to the image plane produces divergent flow, and certain motion discontinuities are also interpretable as divergence. On the other hand, divergence always seems to be associated, in some way or another, with the proximity of an object. The remainder of this section is devoted to an analysis of just what this relationship is, and how it can be most effectively utilized by a navigator.

We consider the image formed by spherical projection of the environment onto a sphere of radius ρ termed the *image sphere*. The use of spherical projection makes all points in the image geometrically equivalent, which considerably simplifies some of the analyses. Ordinary cameras do not utilize spherical projection, but if the field of view is not too wide, the approximation is reasonably close. Since the distortion is purely geometric in origin, it could be corrected should it prove to be a problem for any particular camera. In an experiment we performed using a camera with a field of view of approximately 20×30 degrees, no correction was necessary in order to obtain usable results.

For the purposes of collision avoidance, we are primarily interested in the components of relative motion parallel to and perpendicular to the sensor. For a given point p on the image sphere, these can be conveniently expressed in terms of local coordinate systems centered on p and on the center of projection c. The locations of points in the environment are expressed in terms of coordinates (X,Y,Z) where (0,0,0) coincides with the center of projection, and the positive Z axis passes through p. Image positions in the neighborhood of p are expressed in terms of coordinates (x,y) where (0,0) coincides with p and the x and y axes with the local projection of the X and Y axes respectively. This is permissible because the image sphere is locally Euclidian. The Euclidian neighborhood of p will be referred to as the *local projective plane*. Since all points in the image are geometrically equivalent under spherical projection, and the divergence is a local operation, all our analysis can be done in terms of these local coordinate systems.

The relevant characteristics of the object projecting to p are its distance from the center of projection and the orientation of the surface. Since the divergence involves only first order derivatives, higher order parameters are not significant. The distance to the object is just its Z coordinate. The orientation of the surface can be described either by the equation for the tangent plane $Z = aX + bY + c$ or by the angles α and β in gradient space where the *tilt*, α, is the angle of the projection of the surface normal on the image sphere, and the *slant*, β is the angle between the surface normal and the Z axis. The following relationships hold between these parameters:

$$a = \frac{\partial Z}{\partial X} = \tan\beta\cos\alpha; \quad b = \frac{\partial Z}{\partial Y} = \tan\beta\sin\alpha; \qquad (1)$$

$$\tan\alpha = \frac{a}{b;} \quad \tan\beta = \sqrt{a^2+b^2}.$$

The scenario described above is shown in Figure 2.

We now define a parameterized, one-dimensional divergence measurement and derive the relationships between this measurement and motion of the sensor relative to objects in the environment. We concentrate on this parameterized version of the divergence both because it contains more information than the usual scalar, and because it is directly calculable from real image sequences.

Let $\mathbf{f}$ be the motion field under spherical projection. We define the *directional divergence* $D_\phi\mathbf{f}$ to be the one-dimensional divergence of $\mathbf{f}$ in the direction ϕ. Symbolically we write

$$D_\phi\mathbf{f} = \frac{\partial f_\phi}{\partial r_\phi} \qquad (2)$$

where f_ϕ is the component of $\mathbf{f}$ in the ϕ direction and r_ϕ is Euclidian distance in direction ϕ. In terms of the local $x-y$ coordinate system, it is easy to show that

$$D_\phi \mathbf{f} = \frac{\partial f_\phi}{\partial r_\phi} = \cos^2\phi \frac{\partial f_x}{\partial x} + \sin^2\phi \frac{\partial f_y}{\partial y} + \sin\phi\cos\phi \left[\frac{\partial f_x}{\partial y} + \frac{\partial f_y}{\partial x} \right]. \tag{3}$$

where ϕ is measured from the x axis.

Clearly, D_ϕ is a linear operator. Thus we can separate the contributions of different motions and consider them independently. First, consider motion of the sensor in a rigid, three-dimensional environment. The motion field at any point p on the image sphere can be represented as the superposition of three separate motion fields: one due to sensor rotation ω, one due to translation v_{perp} perpendicular to the image sphere at point p, and one due to translation v_{par} parallel to the image sphere at p. We will refer to these motion fields as $\mathbf{f}_{rot}$, $\mathbf{f}_{perp}$, and $\mathbf{f}_{par}$ respectively. Hence

$$\mathbf{f} = \mathbf{f}_{rot} + \mathbf{f}_{perp} + \mathbf{f}_{par}, \tag{4}$$

and since D_ϕ is linear

$$D_\phi \mathbf{f} = D_\phi \mathbf{f}_{rot} + D_\phi \mathbf{f}_{perp} + D_\phi \mathbf{f}_{par}. \tag{5}$$

The three preliminary theorems that follow establish the relationship between the various types of motion and the directional divergence.

Theorem 1: If the motion field is entirely due to rotation of the sensor, then the directional divergence is identically zero. That is,

$$D_\phi \mathbf{f}_{rot} \equiv 0. \tag{6}$$

Proof: The proof rests on the fact that the motion field due to rotation is due entirely to the transformational properties of a 2-dimensional sphere, and is independent of the 3-dimensional structure of the environment. Let p be the point at which D_ϕ is being evaluated. Recall that with every point p there is associated a coordinate system (X,Y,Z) centered on the center of projection with the Z axis passing through p. By a coordinate transformation, any rotation of the sensor can be expressed in terms of three orthogonal rotational components, ω_X, ω_Y, and ω_Z. In the local projective plane centered at p, the motion field due to rotation about the X and Y axes is given by $\mathbf{f}_{\omega_{XY}} = \omega_Y \hat{\mathbf{x}} - \omega_X \hat{\mathbf{y}}$. This field is locally constant and all the partial derivatives are zero. Hence there is no contribution to $D_\phi(\mathbf{f}_{rot})$ from ω_X or ω_Y. The motion field due to the rotation about the Z axis is circular about p and is given by $\mathbf{f}_{\omega_Z} = -\omega_\alpha r \hat{\boldsymbol{\theta}}$ in the polar coordinate system centered at p. For this field $\partial f_x / \partial x = \partial f_y / \partial y = 0$ and $\partial f_x / \partial y = -\partial f_y / \partial x = \omega_Z$ and hence $D_\phi(\mathbf{f}_{\omega_Z}) = 0$ by equation (3). Since rotations about all three principle axes produce motion fields with zero divergence, the result follows. The principle importance of this result is to show that that the directional divergence is invariant under rotational motion. This means that its use as a cue is not limited to situations where the sensor is undergoing purely translational motion.

The next result describes the effect of translation perpendicular to the image sphere.

Theorem 2: Let p be a point on the image sphere, and suppose that the sensor is translating along the line from the center of projection c to p with velocity v_{perp} towards a surface S distance Z from c. Then

$$D_\phi(p) = D_\phi \mathbf{f}_{perp} = \frac{v_{perp}}{Z}, \tag{7}$$

independent of ϕ and the orientation of the surface.

Proof: In this situation the motion field is radial about p. Let P be the point on S which projects to p. Consider some other point P_1 on S, at a distance R from the Z axis, and let p_1 be its projection on the image sphere. Let r be the radial distance from p to p_1 in the local projective plane (Figure 3). Then

$$r = \frac{\rho R}{Z} + (\text{second order terms in } R \text{ due to slope of } S) \tag{8}$$

where ρ is the radius of the image sphere. Taking the time derivative gives the radial component of the motion field:

$$f_r = \frac{dr}{dt} = -\frac{\rho R}{Z^2}\frac{dZ}{dt} = \frac{\rho R v_{perp}}{Z^2}. \tag{9}$$

Substituting for R from equation (8) gives

$$f_r = \frac{r v_{perp}}{Z}. \tag{10}$$

Taking the partial of f_r with respect to r gives the directional divergence:

$$D_\phi \mathbf{f}_{perp} = \frac{\partial f_r}{\partial r} = \frac{v_{perp}}{Z} - \frac{r v_{perp}}{Z^2}\frac{\partial Z}{\partial r}. \tag{11}$$

Since we are interested in the divergence at $r=0$ the second term is zero, and the desired result follows.

The effect described above corresponds to the intuitive notion of apparent expansion of an approaching object, which suggested the use of divergence measurements in the first place. However, the parallel component of the translation can also produce an effect, and this must be considered. Translation parallel to an obstacle produces no divergence if the surface is parallel to the direction of translation, in which case locally constant flow is produced. However, if the surface is tilted, the changing depth can produce divergence in the motion field. This effect is described below.

Theorem 3: Suppose that the sensor is translating in a direction parallel to the image sphere at point p with velocity v_{par}. Let P be the point on surface S which projects to p, and let Z be the distance from the center of projection to P. The orientation of S is described by the angles α and β where α is the angle between the direction of translation and the projection of the surface normal at P, and β is the angle between the surface normal at P and the Z axis. Let ϕ be measured from the direction of translation. Then

$$D_\phi \mathbf{f}(p) = D_\phi \mathbf{f}_{par} = \frac{\tan\beta v_{par}}{Z}\left[\cos^2\phi\cos\alpha + \cos\phi\sin\phi\sin\alpha\right] \tag{12}$$

Proof: Without loss of generality, we can take the direction of translation to be the x direction in the local projective plane. The y component of the motion field f_y is then zero in the neighborhood of p and we need consider only the derivatives of f_x. Let P_1 be a point on S near P, and let p_1 be its projection on the image sphere, with X and x representing the coordinates of the points in their respective coordinate systems. Then we have

$$x = \frac{\rho X}{Z} \tag{13}$$

where ρ is the radius of the image sphere. Taking the time derivative we obtain

$$f_x = \frac{dx}{dt} = \frac{\rho}{Z}\frac{dx}{dt} = -\frac{\rho v_{par}}{Z.} \tag{14}$$

Taking the partial with respect to x gives

$$\frac{\partial f_x}{\partial x} = -\frac{\rho v_{par}}{Z^2}\frac{\partial Z}{\partial x} = -\frac{v_{par}}{Z}\frac{\partial Z}{\partial X}. \tag{15}$$

From equation (1) $\partial Z/\partial X = \tan\beta\cos\alpha$. Hence we obtain

$$\frac{\partial f_x}{\partial x} = \frac{\tan\beta v_{par}\cos\alpha}{Z.} \tag{16}$$

In a similar manner, we obtain an expression for $\partial f_x/\partial y$:

$$\frac{\partial f_x}{\partial y} = \frac{\tan\beta v_{par}\sin\alpha}{Z.} \tag{17}$$

Substituting (16) and (17) into (3) gives the desired result. The main importance of this result is to show that when $\cos\phi$ is zero, i.e., when the directional divergence is considered perpendicular to the direction of motion, the effect of parallel translation is zero.

Theorems 1 through 3 describe the relationship between the directional divergence and the motion of the sensor in a rigid, three-dimensional environment. In a less highly constrained situation certain objects in the environment may also be undergoing independent rotational and translational motion. We can use the results already derived to calculate the divergence in this case. Let P be the point which projects to p on the image sphere, and suppose that the object containing P is undergoing independent motion relative to a global environment in which both it and the sensor are moving. This movement can be described by terms specifying the translation of, and the rotation about, P. The translation can be broken into components perpendicular and parallel to the image sphere at p, which can be subtracted from the the corresponding components due to sensor translation to yield the relative translation. The rotation about P can be transformed into a rotation about the center of projection, and a translation parallel to the image sphere at p. These can also be subtracted from the corresponding components due to sensor motion. Hence the effects of independent movement of the sensor and an object can be described by a single set of components which specify the relative motion between the two objects in terms of rotation about the center of projection, and translation perpendicular and parallel to the image sphere at p. Note that the transformation of the rotational motion of the object does not affect the relative perpendicular translation. Thus the relative v_{perp} corresponds to the actual rate at which the distance between the sensor and point P is changing.

Theorem 3 shows that the component of translation parallel to an obstacle can produce divergence in the motion field. Hence the presence of divergence in the flow is not simply equivalent to the presence of an obstacle on a collision course with the sensor. However, other relationships do hold, which make divergence useful as a navigational cue. These relationships are based on the fact that the directional divergence due to $\mathbf{f}_{par}$ is zero when $\cos\phi = 0$, that is, when ϕ is perpendicular to the direction of parallel translation. This follows immediately from Theorem 3. The following two theorems provide the basis for the use of divergence as a cue in obstacle avoidance.

Theorem 4: If the distance between the sensor and an object projecting to point p on the image sphere is decreasing, that is, the perpendicular component of the relative translation is positive at p, then there exists a direction ϕ for which the directional divergence $D_\phi\mathbf{f}(p)$ is positive.

Proof: By Theorem 1, the contribution to D_ϕ from any rotational component is zero. By Theorem 2, the contribution from the perpendicular component of motion is positive and independent of ϕ. If we pick ϕ to be the direction on the local projective plane perpendicular to the parallel component of motion, then the contribution from the parallel component is also zero, and the conclusion follows.

Essentially, this result states that measurements of D_ϕ can be used to detect any object which has a component of motion towards the sensor (and is thus a potential collision hazard). This holds for any combination of rotations and translations of the sensor and the object. Thus any collision hazard can be detected by testing whether there exists some ϕ for which D_ϕ is positive.

An object on a collision course is always indicated by a positive divergence measurement. On the other hand, not all positive divergence indicates an imminent collision. We have seen that translation parallel to a inclined surface can produce divergence flow. In addition, even if an object has a perpendicular component to its motion, it may also have a horizontal component that will prevent a collision. The presence of divergence does, however, indicate the proximity of an object. The fact that all the terms which contribute to the divergence contain the term $1/Z$ suggests that a strong divergence signal indicates a nearby obstacle. The situation is somewhat complicated by the presence of the term $\tan\beta$ in the expression for the divergence due to parallel motion. The term represents the slope of the surface relative to the image sphere, which can be arbitrarily large. The limiting case occurs at a depth discontinuity, where the slope essentially becomes infinite. At such points the divergence signal also becomes infinite, which would seem to limit its value as a proximity indicator. The difficulty is circumvented by making use of the fact that, although the value of the divergence may be arbitrarily high, the total change in the flow is bounded. Thus the divergence due to a relatively distant object can be large, but only over a very short distance in the image. Thus if the divergence is considered over a local region, it can be used as a proximity indicator. The following theorem makes the relationship explicit.

Theorem 5: If $|D_\phi| \geq D > 0$ within a disk of radius r about p, where r is small with respect to the radius ρ of the image sphere, then

$$Z_{\min} \leq \max\left\{ \left|\frac{2v_{perp}}{D}\right|, \left|\frac{\rho v_{par}}{rD}\right| \right\}, \tag{18}$$

where $Z_{\min}$ is the distance to the nearest point projecting into the disk.

Proof: Since there are only two sources of divergence, either v_{perp} or v_{par} must account for at least half the value of D. If the major contribution is from v_{perp}, then from equation (7)

$$\frac{D}{2} \leq |D_\phi f_{perp}| = \left|\frac{v_{perp}}{Z_{\min}}\right|. \tag{19}$$

Hence

$$Z_{\min} \leq \left| \frac{2v_{perp}}{D} \right| \tag{20}$$

If, on the other hand, the major contribution is from v_{par}, then there must be a change Δf_ϕ in the flow of at least $2rD/2$ due to the change in depth across the disk from $Z_{\min}$ to $Z_{\max}$. From equation (14) we have

$$rD \leq \Delta f_\phi = \frac{\rho v_{par}}{Z_{\min}} - \frac{\rho v_{par}}{Z_{\max}}. \tag{21}$$

Now $Z_{\min}$ is maximized, that is the object is at the greatest possible distance, when $Z_{\max}$ is equal to infinity; hence

$$Z_{\min} \leq \frac{\rho v_{par}}{rD.} \tag{22}$$

The result follows from (20) and (22).

Theorem 5 shows that the presence of divergence over any significant area indicates an object which is nearby in terms of the translational velocity of the sensor. The result (18) is valid for arbitrary translations and rotations of the sensor and the object, if the components of relative motion are used. However, it should be noted that rotation of the object results in a relative value of v_{par} proportional to Z, which means that a rotating object can produce divergence even at a large distance. This would not seem to be an important consideration in most practical applications, since most obstacles do not rotate independently.

The above results show that the motion field divergence is a useful but rather equivocal cue if rotational motion is present in the system. If the rotational component of the motion is zero or known, so that the effects can be factored out, somewhat stronger results can be obtained. In particular, if the flow is due purely to translation, then the direction of v_{par} is the same as the direction of the flow. Since this is easily determined, ϕ can be chosen perpendicular to this direction, and D_ϕ used to determine, unequivocally, whether or not v_{perp} is positive. Furthermore, if only translational motion is present, then the object projecting to p is on a collision course if and only if D_ϕ is positive at p, and the flow is zero. In this situation, p corresponds to a focus of expansion. In practice, these results might find considerable application, since it is relatively easy to obtain the rotational components of the motion from simple inertial sensors. It is also possible to robustly obtain the motion parameters of the sensor by purely visual means using a 360 degree field of view [Nels87]. In a system stabilized by either means, the stronger, translational results could be used.

The theoretical results can be summarized as follows. For a sensor with arbitrary rotational and translational motion, any object on a collision course with the sensor will produce a positive directional divergence in the motion field for some angle ϕ; however, not all positive divergence indicates an imminent collision. Conversely, any non-zero directional divergence measurement indicates the presence of an object which is nearby in terms of the relative translational velocities of the sensor and the object; however, not all nearby objects produce divergence in the motion field. As far as theoretical use of information contained in the motion field is concerned, these results are sufficient for low-level obstacle avoidance, although used alone, they could result in maneuvers which are not strictly necessary to avoid collision. The results can be strengthened to eliminate such situations if only translational motion is permitted.

The above results are all fairly simple, and the information available is somewhat limited. The primary value of the analysis lies in the fact that the described divergence measures can actually be obtained, easily and reliably, from real image sequences. Motion field divergence is also a robust cue in the sense that it is stable in the presence of significant amounts of error in the motion field. Since determining the motion field precisely is a difficult task, this is a highly desirable characteristic. Divergence measurements can be obtained which are accurate and reliable enough to be usable for navigation in the manner described. The remainder of the paper is devoted to demonstrating these claims.

IV. DESIGN OF AN OBSTACLE AVOIDANCE SYSTEM

The results obtained above suggest that detection of directional divergence in the motion field could form the basis for a simple visual obstacle avoidance system. In order to investigate this possibility we designed a system to perform navigation using motion field divergence as the sole sensory cue. This system utilizes a sequence of pictures taken by a camera mounted on a moving robot arm. Periodically, flow field divergence is computed from the image sequence and analyzed for evidence of obstacles. This information can then be used to guide the camera away from potential collision hazards. This section describes the design of the image processing system responsible for extracting usable information from a sequence of images.

Motion information was extracted using a differential technique. This approach was chosen because it produces a relatively dense field of values and because it uses only simple local operations. It is thus directly implementable in a highly parallel, connectionist type architecture which would be a consideration in any real system. Correspondence methods, in addition to the problem of identifying correspondences, produce a sparse field of values, which makes calculation of a continuous measurement such as divergence difficult.

Use of a differential method means dealing both with the non-identity of the motion field with the optical flow, and with the aperture problem. The first problem was eliminated by assuming sufficient visual texture in the image to avoid any difficulty. Since the divergence is a highly robust measurement, this is not a very stringent requirement. Ordinary natural objects such as tree bark, broken brick, and human faces were found to have sufficient visual texture to make the method workable. The aperture problem arises from the fact that, at a given point and time, only one component of the optical flow can be recovered, namely the component parallel to the local image gradient. This problem was approached by defining a number of principal axes and processing motion parallel to each axis separately. For each axis, the parallel flow component can be determined at some fraction of the points in the image, namely those where the gradient happens to be parallel to the axis. Information for each axis is contained in a structure referred to as a *partial map*. These structures can be thought of as arrays having the same size as the original image but with unknown values at some points. Thus we obtain a number of partial maps, each of which specifies the component of the flow field parallel to a particular axis at some fraction of the image points. The directional divergence can then be independently calculated for each direction from the partial flow maps and the the results combined for use by the navigation system. This process is described in more detail later.

The differential calculations are most easily implemented when the image motion between frames is incremental. The system was thus designed to utilize sequences where the maximum image flow is less than one pixel per frame. In a system exposed to a large range of flow magnitudes, simultaneous analysis of different resolution versions of the image might be necessary. However, for a sensor moving at a relatively constant speed through a static environment, a single level system proved to be sufficient. The frame rate was chosen to allow an approaching obstacle to be detected in time to avoid it.

An important strategy for reducing the level of noise in the system and obtaining divergence measurements robust enough to be reliable was the principle of accumulation of information over time. The strategy is based on the fact that motion features tend to persist in the same neighborhood for several frames. By combining flow estimates computed for several overlapping blocks of time, the reliability of the flow estimate can be increased. This improvement is a result of averaging partially independent estimates for the same value, which is a well known technique. Most applications of averaging in image processing, however, utilize spatial averaging which has the undesirable side effect of also reducing resolution. Because the differential techniques of flow approximation rely on the small scale variation or texture in the image, any blurring of the image by spatial averaging severely degrades the performance. This does not occur, however, with temporal averaging.

The temporal accumulation of information also improves performance in another way. Note that if only a single motion computation is made, the known points of the partial flow maps will form disjoint sets since the gradient at a point has a unique direction. However, if information is combined over several time frames, the number of known points in each map increases and the sets can overlap, especially if the image contains fine grained visual textures which cause the gradient direction at a point to change rapidly. Thus not only is the information more accurate, but there is more of it. In our experiments, the introduction of temporal accumulation of information dramatically increased the reliability of our system.

The process of converting an image sequence to a retinal hazard map involves the following steps.

1. Creation of partial maps on the basis of gradient direction, and computation of spatial and temporal derivatives.
2. Computation of the parallel component of optical flow for each partial map.
3. Temporal accumulation of information by combining separately computed estimates of the flow at several closely spaced times.
4. Approximation of the directional divergence from partial flow maps in each principal direction.
5. Combination of directional divergence maps to produce a retinal hazard map.

We now give a more detailed description of the above steps.

The images acquired by the camera were 512×512 pixels in size. Before processing, these were reduced using averaging by a factor of four to 128×128. This was done primarily to reduce the processing and storage requirements to a manageable level, but had the side effect of making it easier to obtain image sequences with subpixel movement between frames. A set of four sequential gray scale images referred to as a *4-sequence* was used as a basis for computing the spatial and temporal derivatives used in extracting flow information.

The first step was to determine gradient information by applying the four principal 3×3 Sobel operators (corresponding to vertical, horizontal, and two diagonal axes) to each of the images in the *4-sequence* (Figure 4). The image points were assigned to partial maps on the basis of the Sobel operator having the greatest response. The gradient information from the four original images was combined to produce directional gradient maps in the following manner. For every pixel, if the gradient direction was the same in each of the four images, then the corresponding pixel in the partial map for that direction was set to the average of these directional gradients. All other pixels in the partial maps were labeled as unknown. The idea was to ensure that the gradient features used in calculating the flow were persistent enough to obtain a reliable value. The result of this is four partial maps corresponding to the vertical, horizontal, and two diagonal axes, each containing all those points at which there is a persistent gradient in the specified direction. More directions could be used in an attempt to increase accuracy, but this would require larger directional operators, which could blur out small textural features, and would also reduce the density of the known points in each partial map.

The time derivative was approximated by averaging the gray scale values over a 3×3 neighborhood in each image in the 4-sequence, and summing the averages using the weight vector (-1,-1,1,1), that is, subtracting the earlier values from the later. A least squares fit, (-3,-1,1,3), is not appropriate in this case because it weighs the distant points more heavily when there is no reason to suppose they are more important. Using the time derivative, the parallel component of the image flow was then computed for each direction by dividing the time derivative by the gradient magnitude at all points where the gradient magnitude was significantly greater than zero. This yielded four partial maps each specifying a particular projection of the flow field at a set of points.

Flow field information was obtained as above for eight overlapping 4-sequences, producing eight registered partial maps for each direction (Figure 5). Information was combined separately for each direction by averaging the known values at every location. If all values at a certain location were unknown, then the resultant value in the combined partial map remained unknown. This temporal accumulation of information both increases the number of known points for each direction and improves the accuracy of the estimates.

The directional divergence was computed for each direction from the combined flow map. This operation made use of large masks, approximately 20×20 pixels, symmetrically divided into positive and negative halves (Figure 6). The number of known points falling inside the mask at each position was counted, and if this was greater than a certain threshold (e.g. 10), then the information was deemed sufficient for an accurate computation of the divergence. The divergence was then calculated by subtracting the average of the values in the negative half of the mask from the average of the values in the positive half. Where there was insufficient information, the pixel was labeled as unknown. The large number of unknown points in the motion maps allowed this process to be efficiently implemented as a generalized Hough transform, which somewhat compensated for the large size of the masks.

The next step was to combine the different one-dimensional divergence maps. This was done disjunctively, that is, the divergence of the combined map was set to the average of the separate divergences if at least one of the separate maps had a known value at that position. It might be supposed that a more robust procedure would be to combine the separate divergence maps conjunctively, that is, to require supporting evidence from at least two directions. In practice, however, this did not yield good results, primarily because texture edges in real images often possess a local correlation which prevents recovery of information for more than one gradient direction.

The combined divergence map was then converted into a navigation hazard map as follows. Regions where a divergence value had been computed which had a magnitude less than the expected error of the computation were labeled "safe", i.e., no detectable divergence. Regions with a divergence value of magnitude greater than the expected error were labeled "danger" in proportion to the magnitude of the divergence. (In this experiment regions of high negative divergence were also labeled as dangerous since they probably indicate the proximity of an obstacle.) Finally regions where no divergence could be computed were labeled "caution".

This hazard map was used by a simple navigation program to guide the robot bearing the camera. Basically, the field of view was divided into a 3×3 grid and the robot directed towards the square having the greatest "safe" and the smallest "danger" and "caution" areas. More complex schemes could easily be devised, but this proved sufficient to demonstrate the viability of the technique.

The computation of the directional divergence described above can be implemented in a highly parallel, layered connectionist architecture using only local connections. In each layer there is one processor per pixel which receives information from a local neighborhood in the layer below. Information passes through the layers is one direction, so there is no problem with settling times. Time averaging and calculation of time derivatives can be performed by means of delay buffers. Figure 7 shows a schematic diagram of one connectionist architecture which could be used. The squares represent data storage in the form of partial images, and the circles represent operations by which the partial images are transformed into others. Stacks of squares represent delay buffered data used for the temporal accumulation of information. The arrows show the data dependencies and the flow of information through the system.

V. IMPLEMENTATION AND TESTING

The system described above was investigated using an American Robot Merlin robot arm having six rotational joints and a sphere of operation approximately two meters in diameter. A Sony DC-37 CCD miniature television camera with a 16mm focal length lens was attached to the arm. The arm was used to move the camera through a three dimensional environment containing various obstacles. Some attention was necessary to ensure that the obstacles in the environment did not interfere with the motion of parts of the arm distant from the camera, since solving the motion planning problem for the entire robot was not a goal of this study. In general however, the arrangement provides a flexible method for investigating the problems of visual navigation. In order to avoid becoming entangled in the so called piano mover's problem, experiments were carried out in relatively uncluttered environments, i.e., ones in which obstacles can be avoided by changing course or moving aside.

Since the differentially based detection of divergence depends on the presence of small scale variation in the image gray level, both the obstacle and the background were chosen to provide strong visual texture. The sort of visual texture needed is present in most natural objects, and we ran our tests using obstacles such as broken cinder bricks and pieces of tree bark with satisfactory results. It is a curious property of our method that because of the requirement for complex visual textures, it would not be expected to work well in the blocks world.

As described above, the unit of input to the system is a sequence of 12 images, that is 8 overlapping 4-sequences. In the scale chosen for the demonstration, this corresponds to a camera translation of one to two inches. The image processing was performed on a VAX 11/785 computer, and the time needed to extract the divergence from the basic sequence of 12 128×128 images (reduced from 512×512) was on the order of several minutes. This is not quite fast enough for real-time operation; however, the algorithms were deliberately designed to be implementable on a fine grain parallel architecture or neural net, and would be extremely fast on such a system.

We first tested the response of the system to individual translational and rotational movements to see whether the output of the divergence detector corresponded to theoretical predictions. This was essentially a test of the ability of the detector to respond to divergence in actual image sequences in a reliable and robust manner. The environment for these tests consisted of a distant textured background (actually a piece of cloth with a complicated Paisley print), and an obstacle in the foreground (variously a piece of tree bark or a broken cinder block).

The first test consisted of moving the camera directly forward towards the obstacle. Theoretically, the obstacle should display a greater divergence than the background and should be detectable on that basis. Figure P1 shows the view seen by the camera at the beginning of motion towards the obstacle, in this case a piece of bark. In sequence of 12 images input to the system, the bark expands by about 10%, corresponding to a camera motion of about one and one half inches. The change between frames is almost indistinguishable to the eye when two successive images are viewed side by side. However, it is interesting to note that if the two images are flashed alternately on the same screen, a distinct difference is noted, and the bark seems to pop back and forth. Figure P2 shows the hazard map generated by motion directly towards the obstacle. White areas correspond to regions labeled "danger", gray to regions labeled "caution", and black to "safe" regions. The bark stands out clearly as an obstacle. The gray border around the image is an artifact of the lack of any motion information outside of the image.

In the next test, the camera was moved parallel to the obstacle. In this case, according to our theory, the flow discontinuities at the boundaries of the obstacle should stand out. Figure P3 shows the results of this test for the piece of bark. As expected, the boundaries stand out clearly.

We next tested the principle of invariance under rotation. Recall that the divergence was invariant under rotation of the camera. Thus image sequence resulting from pan, tilt, or roll of the camera should show no divergence. Figure P4 shows the hazard map resulting from a camera pan of about two degrees corresponding to an image motion of approximately one half pixel per frame. As expected, little or no divergence is detected. Figure P5 shows the hazard map resulting from a camera roll of about 10 degrees. The flow field for this motion is a circular whirlpool about the center of the image. Mathematically, the divergence for this pattern is zero, but the interaction of the components is more critical than in the previous case. There are a few small regions of spurious divergence reported for this sequence, but basically, the map is clear as expected.

The above tests indicated that our divergence detector worked fairly robustly in detecting the qualitative divergence features predicted by the theoretical analysis. The information should thus be usable for navigational purposes. We tested this by implementing a very simple navigation algorithm. Basically, the camera moves straight ahead for the distance necessary to obtain a sequence of 12 images. This sequence is then analyzed to produce a hazard map. This map is then divided into sections by a 3×3 grid. Each section is assigned a hazard index by adding the number of "danger" points to 1/10 the number of "caution" points. The direction of movement is then changed to point towards the center of the section with the lowest hazard index, and another sequence of 12 images is acquired.

Figures N1-N10 show the results of a test run where the system navigated through the gap between two pieces of bark. Figures N1-N10 (a) show the scene viewed by the camera at the beginning of each step, and Figures N1-N10 (b) show the hazard map at the end of each step. The obstacles were large enough to prevent the robot from seeing around them, and the robot was prevented from navigating up and over the obstacles by allowing only left-right directional changes. This forced the system to find the gap. Figure 8 shows the path followed by the camera as it approached the obstacles. Note how the robot switched course back and forth to in order to keep heading for the gap between the obstacles.

Thus we observe that even a very simple system, without any elaborate spatial reasoning abilities, can exhibit significant visual navigational skill. There is certainly room for improvement in the navigation algorithm. For example, the hazard map could be updated more frequently, rather than at the end of disjoint 12-image sequences, which would allow curves to be negotiated more smoothly. Since the divergence is unaffected by camera rotation, changing the direction of translation continuously rather than at discrete points would pose no problem. The system could also be instructed to handle situations where no free path is visible by making a large turn or backing up and turning. The problem could also be approached by expanding the effective field of view with additional cameras. This would permit the robot to sense obstacles to the sides as well as straight ahead.

Overall, the system displayed a high degree of robustness, as would be expected with a qualitative approach. The results are relatively insensitive to noise in the images. This is evident from the fact that the procedure works with real images taken from a robot mounted television camera. In fact, replacing one or two images in the sequence with other images having no relation to the scene at hand affects the results hardly at all. (This was done accidentally during the course of the experiment). The internal thresholds used to determine whether a value represents information or noise were determined a priori from rough geometric error analysis. Thus no extensive testing was necessary in order to determine parameter values. In general, the results seem to be insensitive to the details of the implementation as long as the general outlines of the procedure are followed. This is an encouraging property.

VI. CONCLUSIONS

We have argued that the flow field divergence represents a qualitative measurement which can be used as a powerful cue in the process of obstacle avoidance during visual navigation. We have further demonstrated that the flow field divergence can be computed robustly from real image sequences using relatively simple operations. Two basic principles which facilitated this task were directional decomposition of the problem into multiple one-dimensional domains, and temporal accumulation of information which permitted the use of many-image sequences. Finally, we presented a practical demonstration in which image flow divergence was used to maneuver a vehicle between obstacles in a real world environment.

In a more general sense, we believe this work supports two claims. First, that many problems in vision can be solved without the sort of exact, quantitative reconstruction of the world which has proven so difficult to obtain. Obstacle avoidance using a qualitative measure of flow field divergence is one example of such a problem. Other simple examples include visual stabilization and target tracking. It is also possible that more complex problems such as bin picking, and high level navigation tasks such as docking, can be performed without reference to an explicit quantitative representation of the world through a direct association of patterns with actions. We are currently pursuing research in this area. Second, in the study of autonomous systems, the development of general and robust solutions to specific tasks is at least as important as the development of mathematical solutions to general problems which hold under specific theoretical conditions. Specifically, for many problems, practical solutions of the latter kind may prove infeasible or at least inappropriate, because of the difficulty of mathematically characterizing real world environments in a useful way. This project has approached the use of motion sequences from the perspective of the particular practical problem of obstacle avoidance by a moving sensor rather than in the general, theoretical manner which has characterized much of the previous work on motion and image flow.

This research has applications in the visual guidance of industrial robots, and in the navigation of autonomous aircraft or submarine vehicles. The basic premise is that once a basic navigational procedure for the avoidance of obstacles has been established, higher level directives (such as "try to go north") can be imposed on the system since, in most cases, considerable freedom exists in choosing a path between obstacles.

REFERENCES

[Adiv85] G. Adiv, Inherent ambiguities in recovering 3-D motion and structure from a noisy flow field, Proc. 1985 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 70-77.

[Anan85] P. Anandan and R. Weiss, Introducing a smoothness constraint in a matching approach for the computation of optical flow fields, Proc. Third Workshop on Computer Vision: Representation and Control, 1985, 186-194.

[Barn80] S. Barnard and W. Thompson, Disparity analysis of images, *IEEE Trans. PAMI* **2**, 1980, 333-340.

[Broo86] R. A. Brooks, Achieving artificial intelligence through building robots. A. I. Memo 899, Massachusetts Institute of Technology Artificial Intelligence Laboratory, May, 1986.

[Horn81] B.K.P. Horn and B.G. Schunk, Determining optical flow, ***Artificial Intelligence*** **17**, 1981, 185-204.

[Lawt83] D.T. Lawton, Processing translational motion sequences, ***Computer Vision, Graphics, and Image Processing*** **22**, 1983, 116–144.

[Long81] H.C. Longuet-Higgins, A computer algorithm for reconstructing a scene from two projections, *Nature* **293**, 1981.

[Luca85] B.D. Lucas and T. Kanade, Optical navigation by the method of differences, Proc. International Joint Conference on Artificial Intelligence, 1985, 981–984.

[Marr82] D. Marr, *Vision,* W. H. Freeman & Company, San Fransisco, 1982

[Nels87] R.C. Nelson and J. Aloimonos, Finding Motion Parameters from Spherical Flow Fields, University of Maryland Computer Science Department TR-1840, April 1987.

[Reic76] W. Reichardt and T. Poggio, Visual control of orientation behavior in the fly. Part I, *Quarterly Reviews of Biophysics* **9**, 3, 1976, 311-375.

[Thom86] W.B. Thompson and J. K. Kearney, Inexact vision, Workshop on Motion, Representation, and Analysis, May 1986, 15–22.

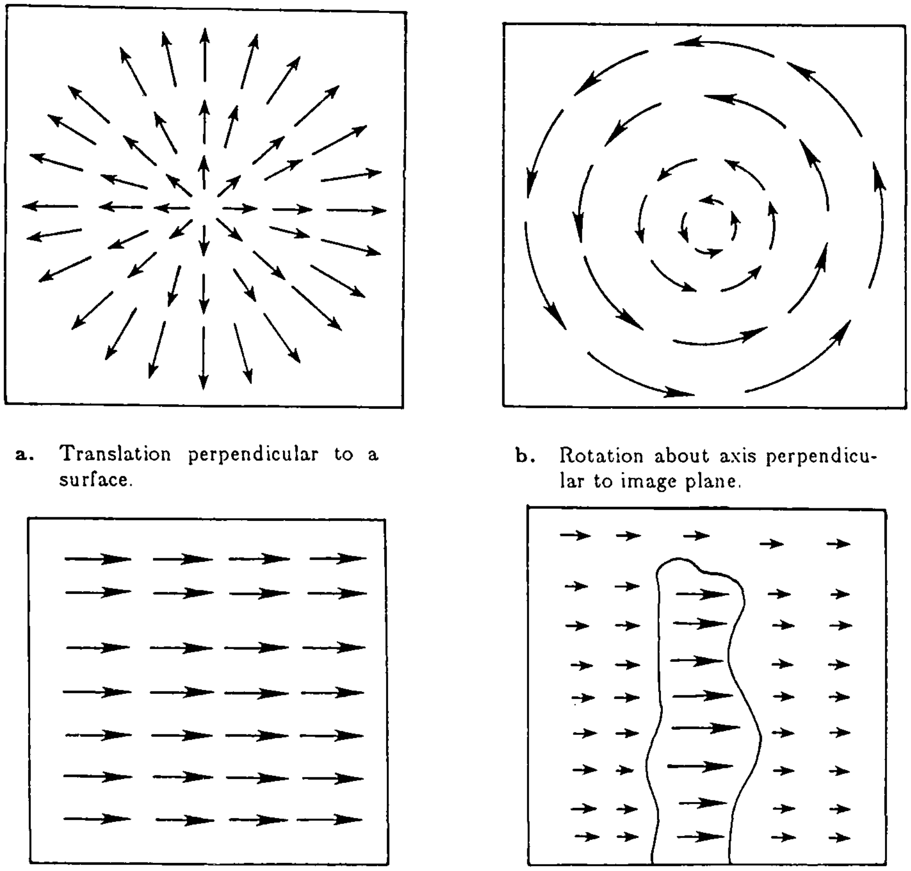
[Tsai84] R.Y. Tsai and T.S. Huang, Uniqueness and estimation of three-dimensional motion parameters of rigid objects with curved surfaces, *IEEE Trans. PAMI* **6**, 1984, 13-27.

[Ullm79] S. Ullman, The interpretation of structure from motion, *Proceedings of the Royal Society of London* **B** **203**, 1979, 405–426.

[Ullm81] S. Ullman, Analysis of visual motion by biological and computer systems, *IEEE Computer* **14**, 1981, 57–69.

[Verr87] A. Verri and T. Poggio, Against quantitative optical flow, International Conference on Computer Vision, June 1987, 171-180.

[Yasu] Y. Yasumoto, Experiments in estimation of 3-D motion parameters from a sequence of image frames, Proc. 1985 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 560–571.



a. Translation perpendicular to a surface.

b. Rotation about axis perpendicular to image plane.

c. Translation parallel to a surface at a constant distance.

d. Translation parallel to an obstacle in front of a more distant background.

Figure 1: Examples of motion fields.

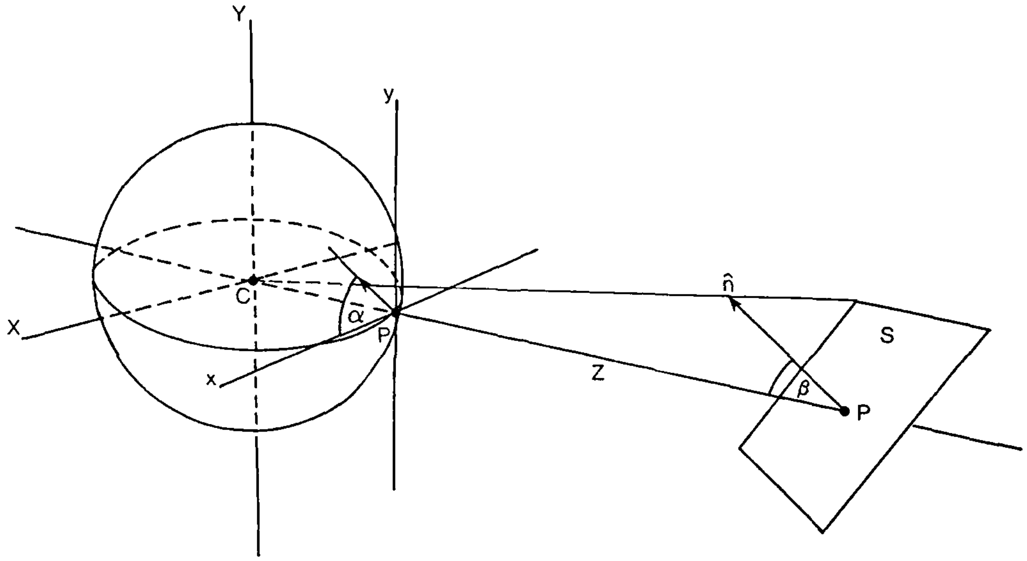


Figure 2: Spherical projection geometry and local coordinate systems.

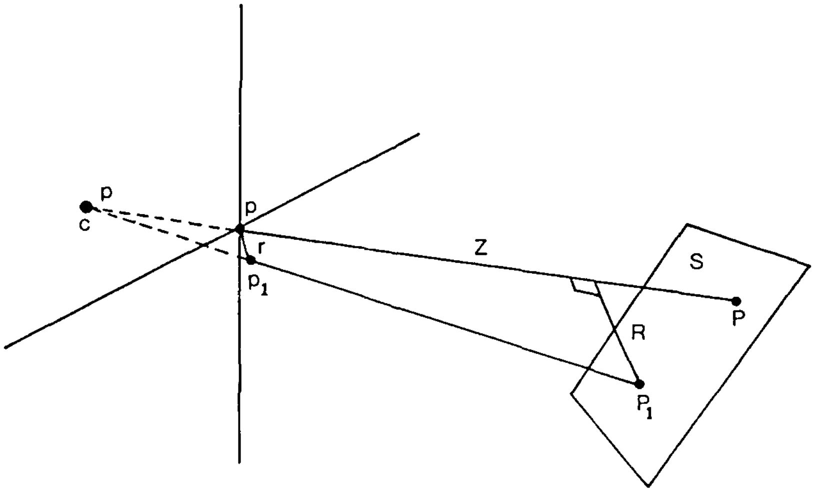


Figure 3: Projection of two points on surface S onto the local projective plane.

$$\begin{matrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{matrix} \qquad \begin{matrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{matrix} \qquad \begin{matrix} -1 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & 1 \end{matrix} \qquad \begin{matrix} 0 & 1 & 1 \\ -1 & 0 & 1 \\ -1 & -1 & 0 \end{matrix}$$

Figure 4: Principal 3x3 Sobel operators

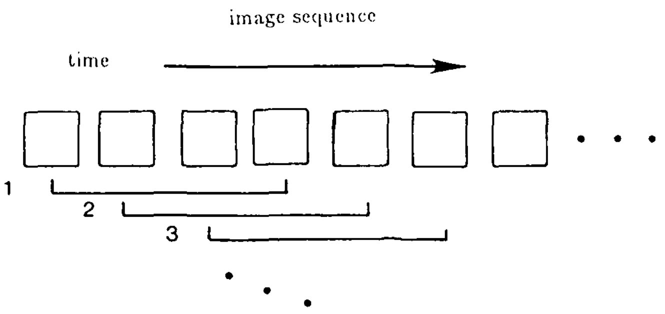


Figure 5: Formation of overlapping 4-sequences.

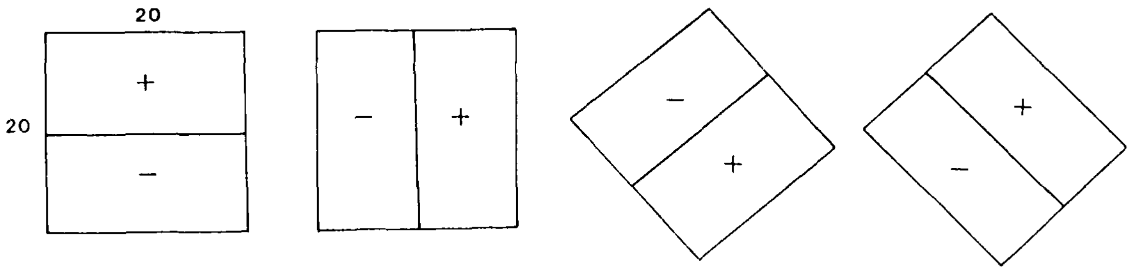


Figure 6: Large masks used for detection of divergence.

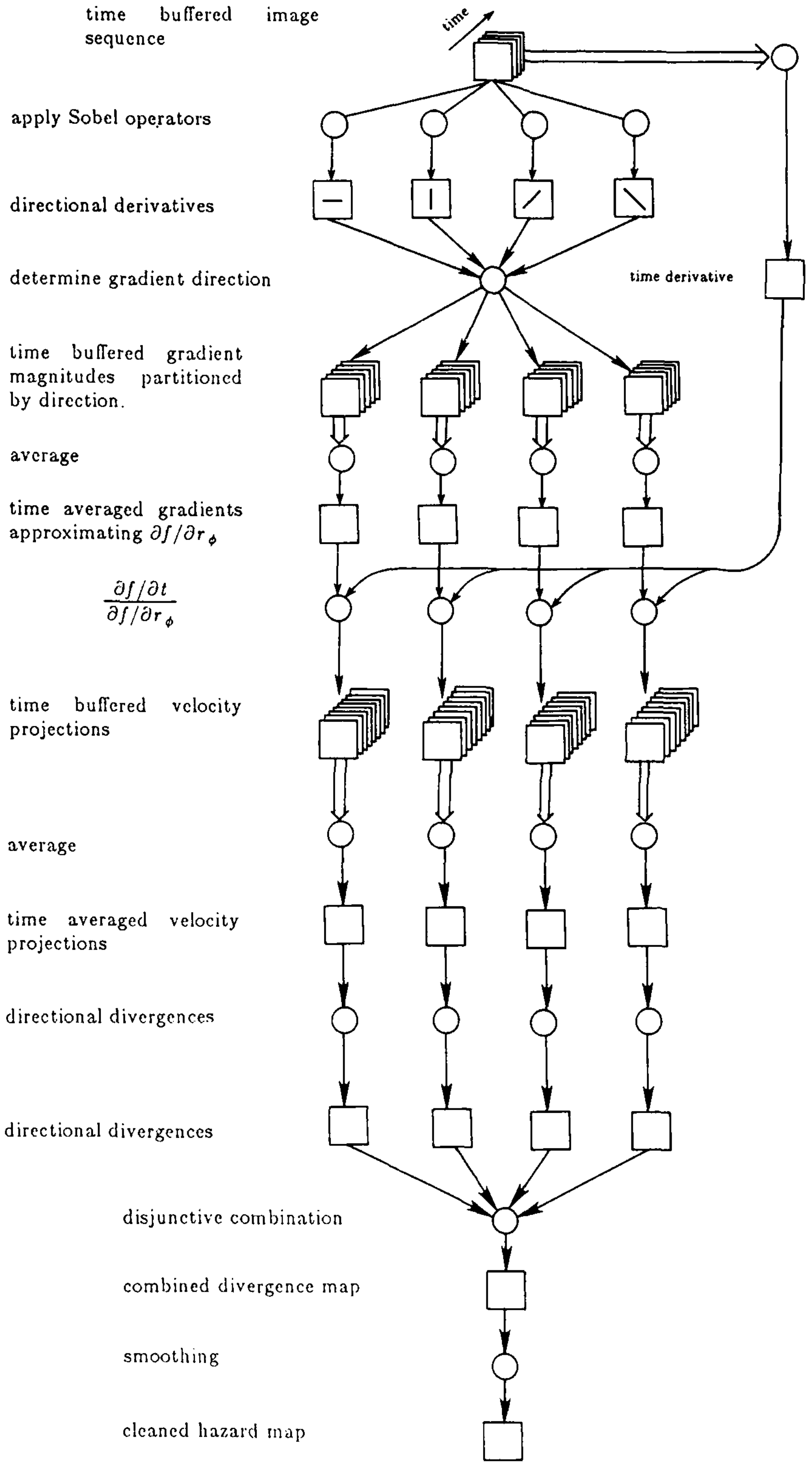

time buffered image sequence
time
apply Sobel operators
directional derivatives
determine gradient direction
time derivative
time buffered gradient magnitudes partitioned by direction.
average
time averaged gradients approximating $\partial f/\partial r_\phi$
$\frac{\partial f/\partial t}{\partial f/\partial r_\phi}$
time buffered velocity projections
average
time averaged velocity projections
directional divergences
directional divergences
disjunctive combination
combined divergence map
smoothing
cleaned hazard map

Figure 7: Computation of the directional divergence and hazard map.

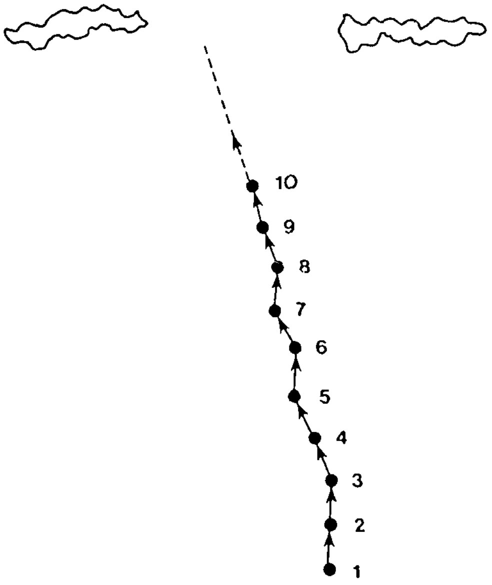


Figure 8: Path of robot navigating between a pair of obstacles.

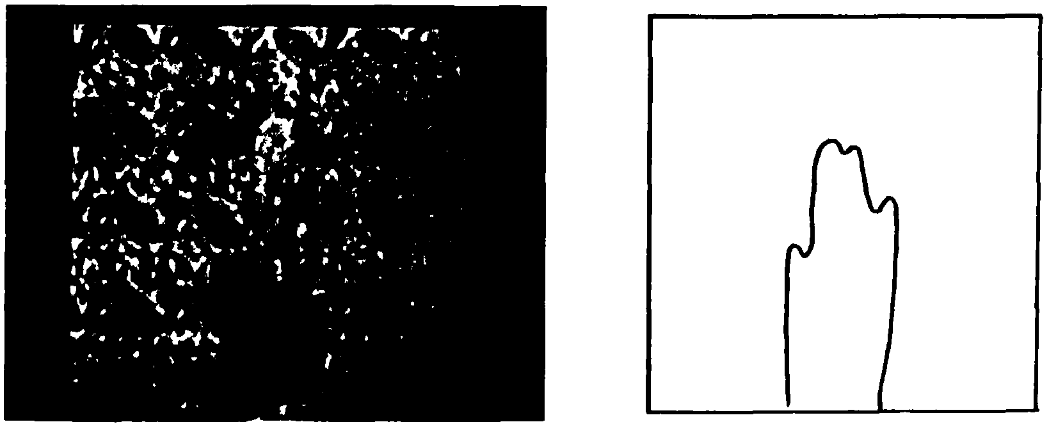


Figure P1 Obstacle (piece of bark) against textured background

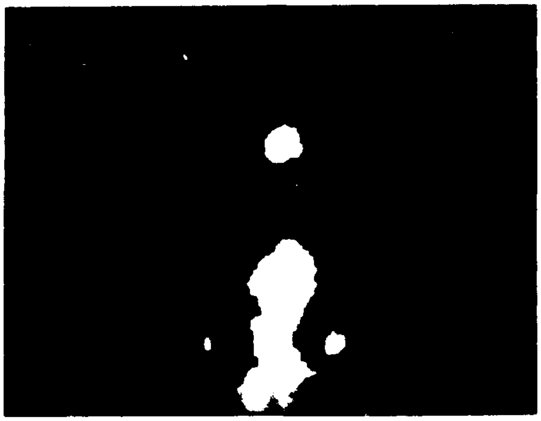
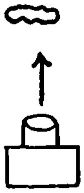


Figure P2: Perpendicular translation.



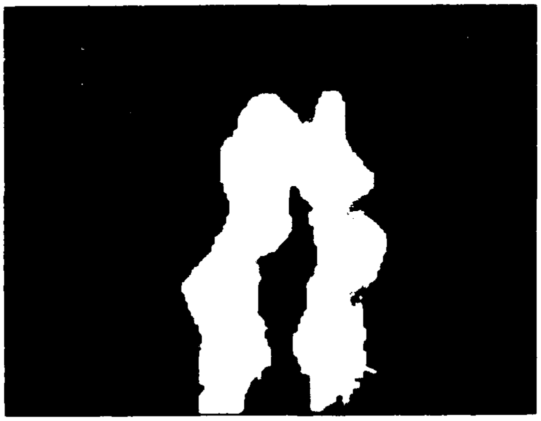
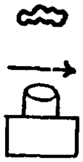


Figure P3: Parallel translation.



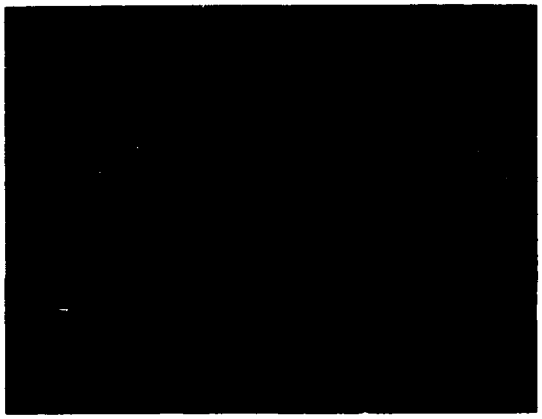
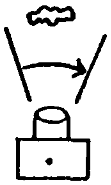


Figure P4: Rotation (pan).



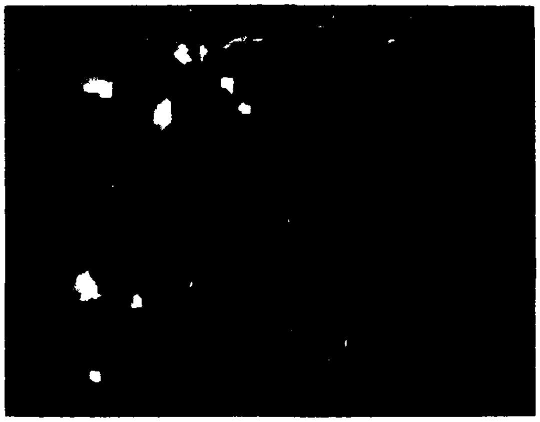
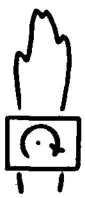


Figure P5: Rotation (roll).



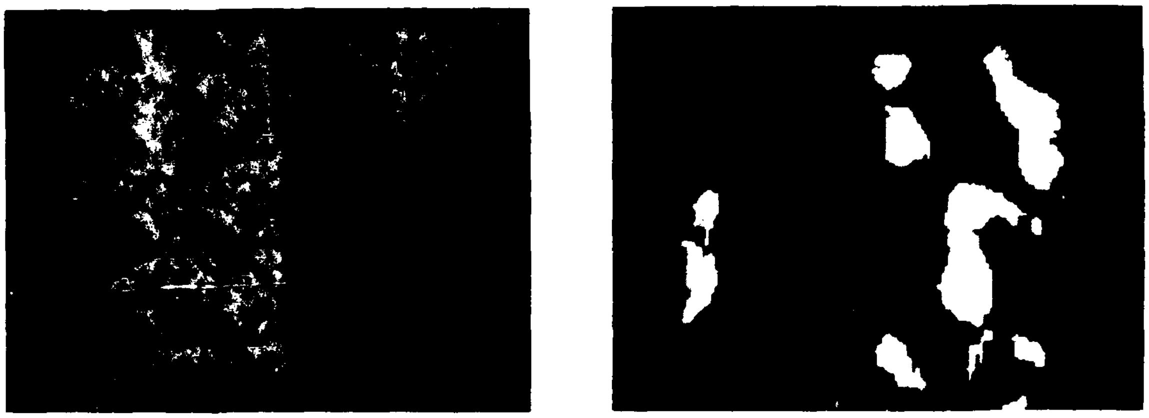


Figure N1

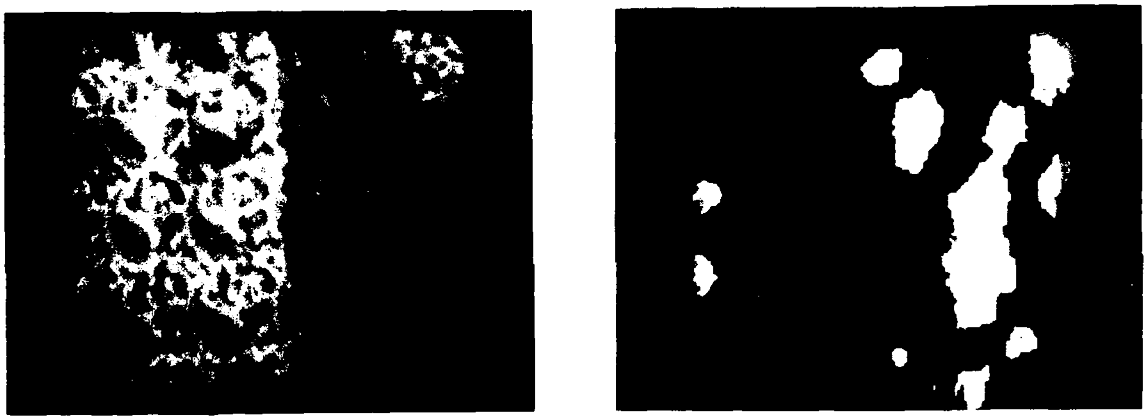
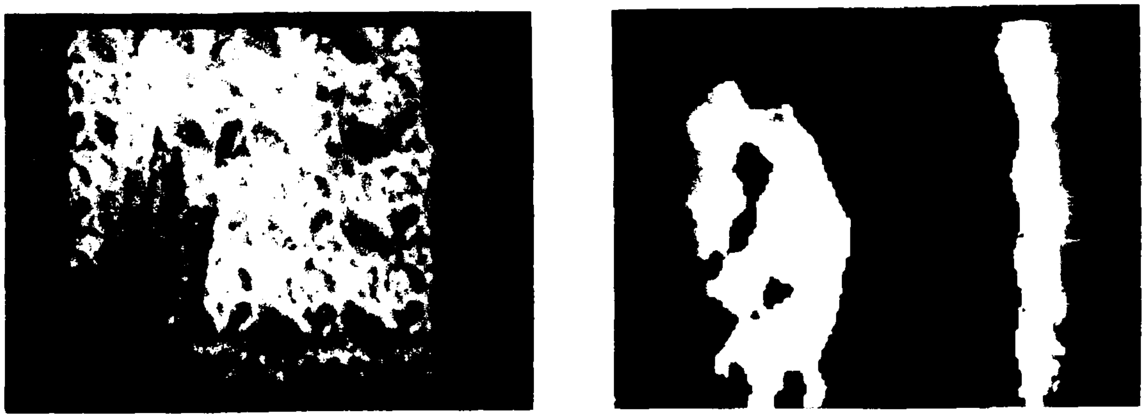


Figure N2



a Figure N3 b

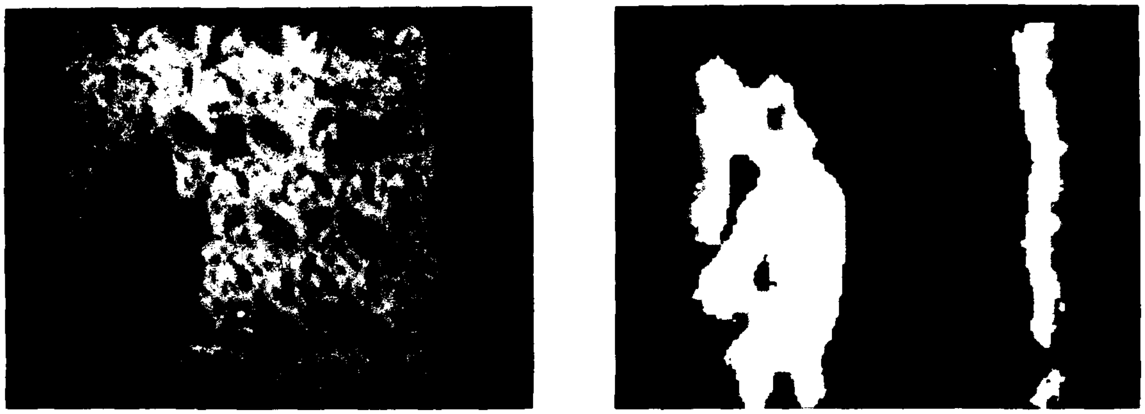


Figure N4

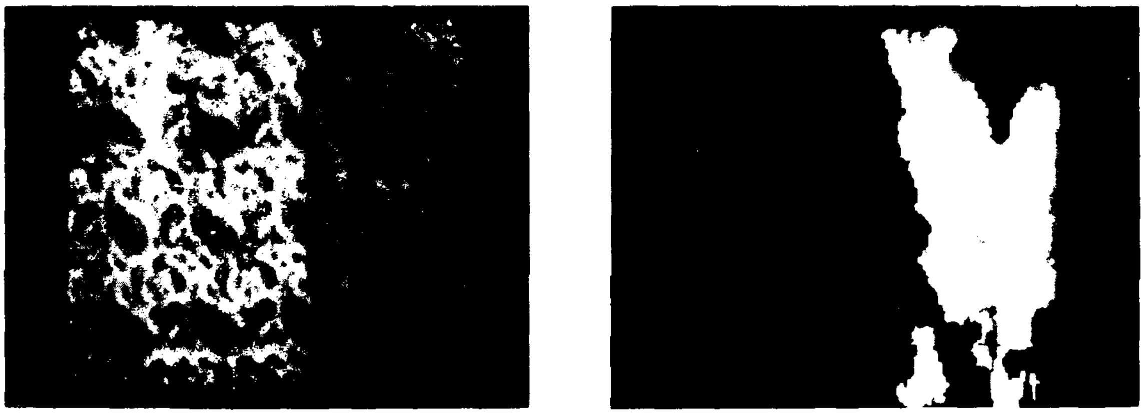
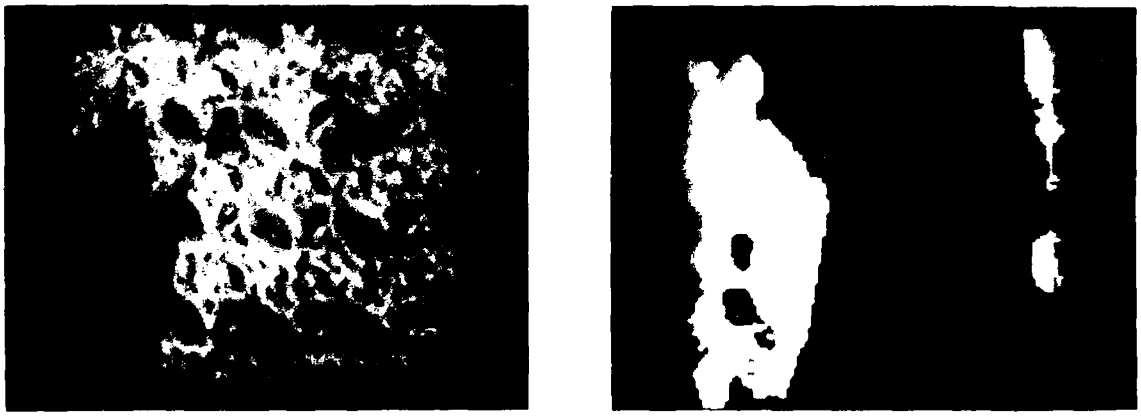


Figure N5



a Figure N6 b

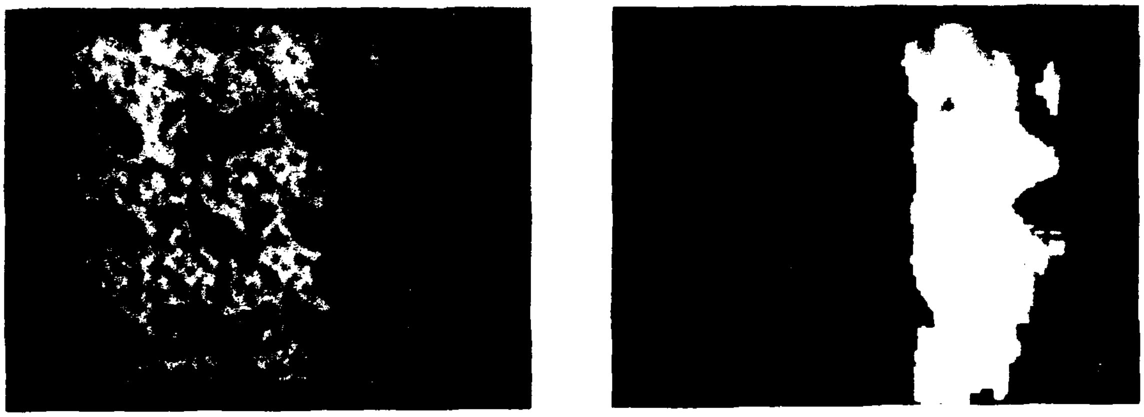
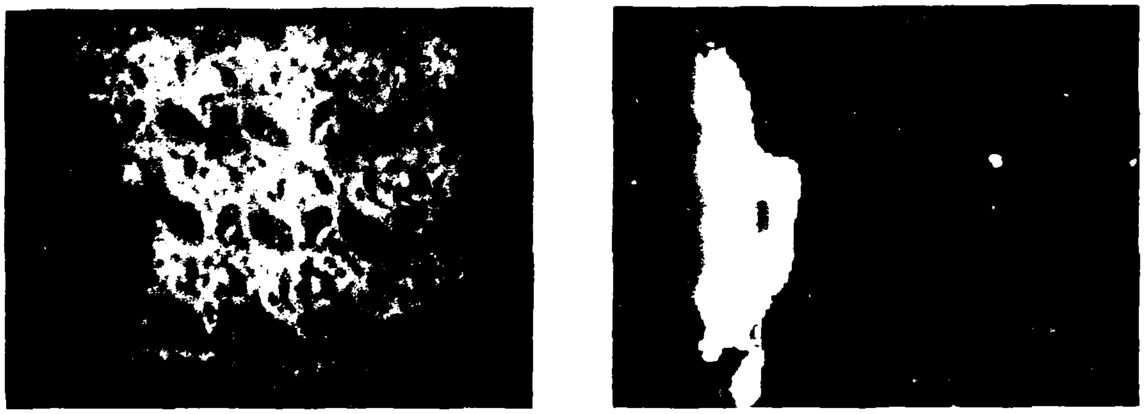


Figure N7



a

Figure N8

b

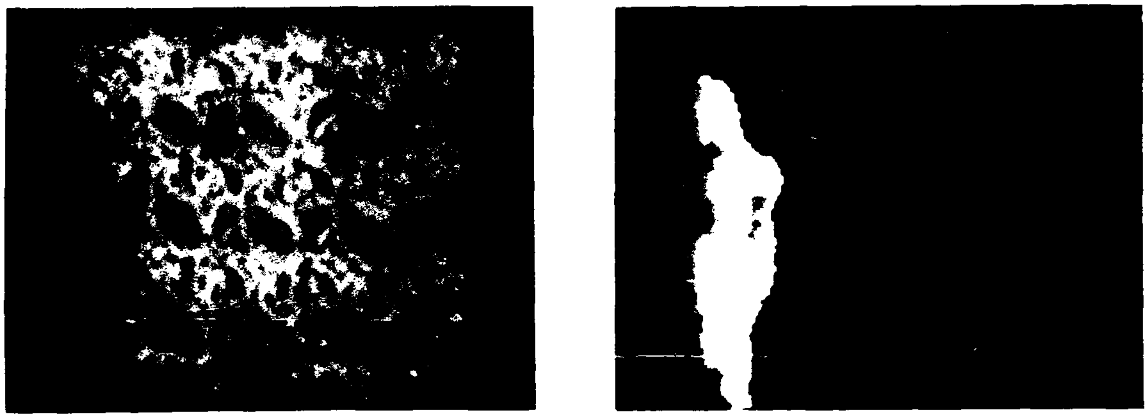
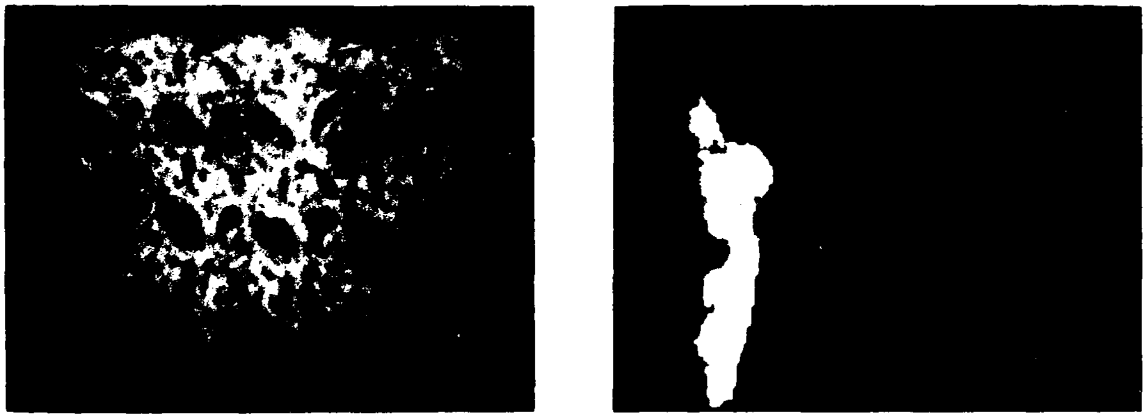


Figure N9



a Figure N10 b

AN OPERATIONAL PERCEPTION SYSTEM FOR CROSS-COUNTRY NAVIGATION*

Michael J. Daily, John G. Harris, and Kurt Reiser

Hughes Artificial Intelligence Center
Calabasas, CA. 91302

Abstract

We present an operational perception system for cross-country navigation which has been verified in both simulated and real world environments. Range data from a laser range scanner is transformed into an alternate representation called the Cartesian Elevation Map (CEM). A detailed vehicle model operates on the CEM to produce traversability information along selected trajectories. This information supports a real-time reflexive planning system. During the months of July and August, 1987 we successfully demonstrated our obstacle detection and avoidance algorithms on board the Autonomous Land Vehicle at Martin Marietta Aerospace Corporation in Denver. We also propose enhancements to our current operational system.

1. INTRODUCTION

Think for a moment of the perceptual differences between the indoor world and natural outdoor terrain. A perception system which extracts features of rectalinear surfaces for recognition purposes may be well suited for identification of man made objects in an indoor environment, but the same system would fail miserably when faced with an outdoor scene. The distinguishing features of indoor objects such as chairs, tables, carpets, and walls have specific meanings in the indoor environment where they are found. For example, the vertical edges of a chair would be perceived as an obstacle to mobility, whereas the color of the carpet may be a clue to the current location. Attempting to extract features, identify objects, and build a description of the environment in order to navigate is a practical approach for indoor mobile robots since in most cases the intraversable regions are occupied by vertical obstacles and the floor is flat. However, meaningful objects in natural terrain are much more difficult to characterize. Specific features such as slope, three dimensional edges (e.g. abrupt changes in vertical height of the terrain), or color of specific regions may or may not signal obstacles and are difficult to extract and combine to form reliable descriptions of the terrain. For example, trees can have widely varying shapes, colors, and sizes making tree recognition in itself a formidable problem. With current capabilities, collecting object descriptions in such a manner provides at best a poor understanding of the environment and is not sufficient for safe vehicle navigation.

In the context of cross-country navigation, a more satisfactory definition of obstacles is any region a particular vehicle cannot traverse. The definition of obstacles depends upon the mobility characteristics of the vehicle. Therefore, a natural approach to perception for cross-country navigation models the vehicle's interaction with a three dimensional representation of the sensed terrain at locations of interest to determine traversability. In this paper we describe such an operational perception system for navigation in complex outdoor terrain with the Autonomous Land Vehicle (ALV). The ALV is an eight wheeled vehicle approximately 5 meters long and 3 meters wide with a minimum clearance of roughly 22 centimeters (Lowrie et. al. 1985). Three dimensional data is obtained from a laser range scanner developed by the Environmental Research Institute of Michigan (ERIM) (Zuk and Delleva 1983) (Sampson 1987). The combination of the mobility characteristics of the ALV and the ERIM scanner are adequate to perceive the scale and type of cross-country terrain shown in Figure 1. We also briefly discuss the hierarchy for the supporting perception and planning systems and present results from an actual cross-country experiment at Martin Marietta Aerospace Corporation (MMAC) in Denver during August, 1987. Finally, we outline several new enhancements which have been tested in simulation.

2. CROSS-COUNTRY PERCEPTION SYSTEM

In our recent experiments, cross country navigation is carried out by a hierarchical control system (Daily et al. 1988). At the highest level of the hierarchy resides a map based planner providing goal information obtained from digital terrain maps to the next lower level in the hierarchy, the local planner. At the lowest level of the hierarchy is the reflexive planner which performs real time control of the vehicle. The local planner ensures that the reflexive planner maintains progress toward subgoals. A similarly hierarchical, multi layered perception system interacts with each corresponding level of the planning hierarchy. In this paper, we are concerned only with the lowest level of the perception system. In this level, sensing and processing units, called "virtual sensors", detect specific environmental features and relay information about features to the reflexive

* This research was supported in part by the Defense Advanced Research Projects Agency under contract DACA76-85-C-0007

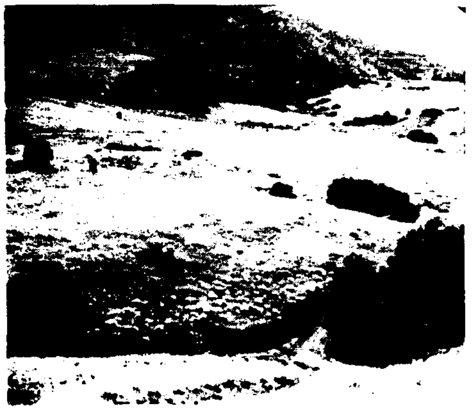


Figure 1. Typical cross-country terrain for the ALV.

behaviors in the lowest level of the planning system. Virtual sensors provide information by combining physical sensor output with appropriate processing algorithms in a manner transparent to the requesting behavior. In this way, the vehicle reacts to obstacles in its environment at real time speeds in a manner consistent with its planned goals.

Virtual sensors process data output from physical sensors on board the vehicle. The ALV is equipped with an ERIM laser range scanner which measures the distance along the line of sight to the nearest object. Distance (or "range") is actually computed by measuring, at specified angular intervals, the phase shift between the active and reflected laser signals. In the range scanner used on the ALV, the maximum detectable phase shift corresponds to a physical distance of 64 feet. Additionally, since phase shifts are measured modulo 2π, all recorded distances are modulo 64 feet. The range resolution of the scanner is 3 inches; it has a horizontal field of view of 80° and vertical field of view of 30°. The actual range image is a rectangular array of 64 rows by 256 columns containing 8-bit range values. Figure 2 shows a sample range image with range values encoded as brightness (i.e. darker values are closer to the sensor). The scene contains a large, deep ravine with trees on the left and gently sloping grassy fields on the right. Though the sensor supplies information which inherently contains surface geometries, reclamation of surface description is difficult in complex outdoor imagery. In the following sections we describe methods for transforming range images into a simpler representation and the use of this representation for vehicle navigation.

Figure 2. Range image of cross-country terrain.

2.1 CARTESIAN ELEVATION MAP

There are numerous approaches for detecting obstacles in range imagery which perform calculations in the angle/angle coordinate system of the range sensor (Tseng et al. 1986) (Daily, Harris, Reiser 1987) (Hebert and Kanade 1985). However, we have encountered several problems with conventional image processing methods which operate in the plane of the range image. For example, since the sensor moves the scanning ray over the environment at uniformly incremented angles, areas near the sensor are much more densely sampled than those areas farther away from the scanner. A small neighborhood of pixels at the top of the range image typically represents an area in three-space two orders of magnitude larger than the same size neighborhood at the bottom of the image. Consequently, convolving a range image with a mask of constant size is equivalent to convolving a Cartesian representation of the same image with a mask which enlarges and warps as it moves away from the scanner. Thus, features of uniform size in the real world cannot easily be detected with mask operators of constant size in range images. A second difficulty arises when attempting to fuse scans of the same scene taken from different viewing positions. Since range images have fine resolution for close objects and coarse resolution for distant objects, small changes in viewing position create large changes in the scale and resolution of scene objects. Consequently, fusion of multiple scans is extremely problematic in the angle/angle space of range images.

The Cartesian Elevation Map (CEM) is an alternate representation for range information in which data from the viewer centered coordinate system of a range sensor is transformed into a Cartesian $z(x, y)$ coordinate system. This results in a down-looking, map view representation of terrain which is useful for autonomous navigation. This same representation may be obtained from other depth sensors, such as stereo or sonar.

The first step in converting from laser range data to the CEM is to calculate the x, y, z Cartesian values from each range measurement, $\rho(\theta, \phi)$, throughout the range image using the following transformations:

$$x = \rho(\theta, \phi) sin\theta$$
$$y = \rho(\theta, \phi) cos\theta cos\phi$$
$$z = \rho(\theta, \phi) cos\theta sin\phi$$

x, y and z represent the crossrange, downrange, and elevation coordinates, respectively. In the original range image, ϕ corresponds to the depression angle of a scan ray and θ indicates the sideways deflection of the ray, measured in the plane of ϕ. † Optics in the sensor cause scan rays to diverge as they travel away from the sensor. When a divergent ray falls on an object at some arbitrary angle, it illuminates an

† The coordinate system can be transformed to a true spherical coordinate system by renaming $\phi \rightarrow \theta$, $\theta \rightarrow \phi$, $x \rightarrow z$, and $z \rightarrow x$.

elliptically shaped area often referred to as a laser "footprint". Distance to the footprint is a reflectance weighted average over the illuminated area. Each of the 3D points in the CEM, as derived from the above formulas, denotes the approximate location of the center of a footprint.

Figure 3a shows the actual (x, y) positions of each one of the scanned points for the image in Figure 2 within a 64 foot by 80 foot region in front of the scanner. Elevation data (z values) are present only at these sparse points. As one would expect, the sparsity of scanned points increases with distance from the scanner. We approximate the complex laser footprint scanning procedure as a smoothing followed by a sampling process. Theoretically, if the terrain were sampled well enough (i.e. within the Nyquist rate), then we could accurately reconstruct the original terrain by interpolating a smooth surface between scanned points.

Clearly, there will always be some unknown regions in which there were not enough scanned points to accurately reconstruct a surface. For example, any region outside the field of view of the scanner or in the shadows of tall features in the terrain will be unknown. These areas must be located and excluded from the interpolation process. Figure 3b shows the regions where the density of scanned points was deemed too low to accurately reconstruct a surface.

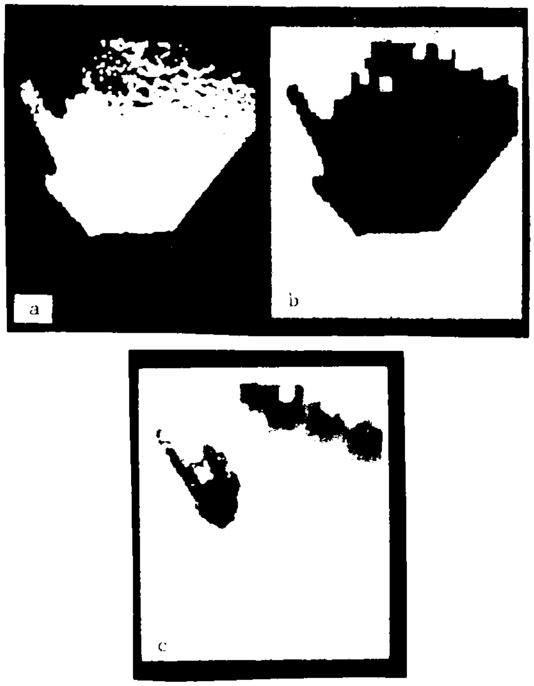


Figure 3. a) Constraints points b) Known areas c) Fully interpolated CEM.

An iterative interpolation algorithm was used to fill in a continuous surface in all regions with a sufficiently dense sampling of points. This algorithm is similar to the standard routines used for interpolating surfaces from sparse stereo data (Grimson 1981) (Terzopoulos 1983). Rather than fitting a thin plate to the surface (i.e. minimizing the quadratic variation), we have found that fitting an elastic membrane is satisfactory. Intuitively, this algorithm is equivalent to fitting a rubber sheet over the set of constraint points and finding the resultant minimal energy surface. Figure 3c depicts the final interpolated CEM, where brighter areas denote higher elevations. Figure 4 shows a 3D plot of the same CEM.

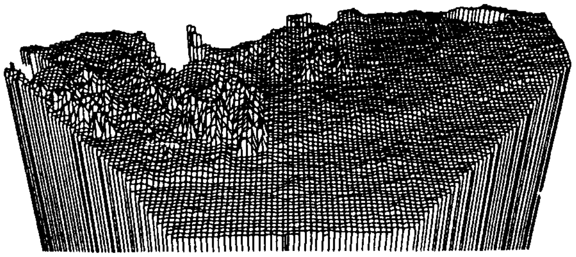


Figure 4. 3D view of CEM.

We have discussed the CEM in terms of the ERIM laser range scanner, but in fact the CEM can be generated from any range sensor. For example, we anticipate using stereo cameras on the ALV to produce range data for CEM construction. The algorithms we use to process the CEM (described in the following sections) can be used without regard to the original range sensor. The following section discusses how the CEM concept is used to fuse data from multiple scans. The CEM has also been used to aid in the combination of data from multiple sensors such as a range sensor and a color camera (Daily et al.1987).

2.2 CEM FUSION

Due to the limited vertical field of view of the ERIM sensor (30 degrees), terrain immediately in front of the sensor will not be visible. For example, the ALV scanner cannot see obstacles directly in front of the vehicle due to the height and tilt of the laser range scanner. The closest scanned ground is approximately 13 feet in front of the vehicle. One way to fill in the unknown area is to use data from previous CEM's. Fusion of CEM's requires exact knowledge of the vehicle displacement in six degrees of freedom between range images. Orientation sensors on board the ALV record the vehicle's heading, pitch, roll, x and y values. A simple estimate of the change in z between two CEM's is calculated by taking the difference of the average z value of a small patch of ground in one CEM and the average z value of the same patch of ground in the other CEM. Provided the output of the orientation sensors and delta z estimates are accurate, CEM's can be reliably fused. We are are currently investigating the recovery of the six degree of freedom motion from sequences of range images.

2.3 USING A VEHICLE MODEL TO FIND OBSTACLES IN THE CEM

The most conclusive way to discern whether a certain patch of terrain is an obstacle is to literally drive a vehicle over the area in question. In lieu of such a time consuming and potentially costly technique, we have developed a relatively sophisticated 3D model of the ALV that we apply to the CEM with various locations and headings. The interaction of this vehicle model and the CEM yields a formal definition of obstacles, thus avoiding *ad hoc*, incomplete definitions.

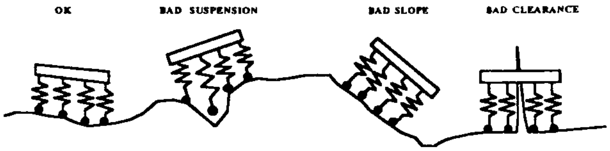


Figure 5. Spring model of vehicle.

The position and orientation of the vehicle at any point in the CEM is determined by minimizing the energy of the suspension springs associated with each wheel of the vehicle. As shown in two dimensions in Figure 5, three types of obstacles are detected with the vehicle model:

Suspension obstacle: detected when any of the eight wheel springs is stretched or compressed beyond its suspension tolerance.

Slope obstacle: occurs when the vehicle tips further than can be safely allowed.

Clearance obstacle: signalled whenever any part of the terrain juts up into the bottom area of the vehicle carriage.

With more processing time, the model can be extended to include constraints such as weight distribution, weather conditions, risk factors, vehicle speed, and dynamics. By formally defining an obstacle in terms of a particular vehicle's characteristics such as clearance and suspension, we can apply this model of the vehicle directly to the sensed terrain to locate traversable regions.

2.4 THE VEHICLE MODEL TRAJECTORY METHOD

The vehicle model allows us to produce a three dimensional configuration space by applying the model at each possible position and heading. In most cases a complete traversability map of the entire sensed area is not needed. Because of the processing time required for the construction of such a traversability map, we have developed a technique for applying the model only at those places necessary to fulfill requests issued from the planner. In the context of virtual sensors (briefly mentioned in Section 2), we use the vehicle model to produce specific information for traversability at fast update rates. This method, called the Vehicle Model Trajectory (VMT) virtual sensor, simply calculates the projected heading of the vehicle at each point along a linear trajectory and applies the model at that heading and location until it either reaches the end of the trajectory or assumes an unstable configuration. The distance travelled (or "safe distance") and the reason for stopping are returned. Figure 6 shows the result of applying the vehicle model at seven linear trajectories in the CEM.

For use on the ALV, we implemented four versions of the Vehicle Model Trajectory virtual sensor, each providing a different level of accuracy and speed. Three parameters may be varied independently: CEM size and resolution, vehicle model accuracy, and sampling frequency of the range image. In applying the vehicle model, computation time can be significantly improved by processing every other pixel along a given trajectory and performing the clearance test at a lower resolution. In our recent experiments (described in the next Section) we typically used a version with one foot per pixel resolution. This version used every other pixel while applying the vehicle model in the CEM.

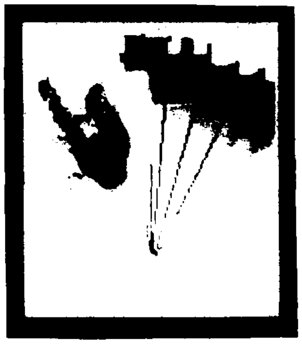


Figure 6. Vehicle Model Trajectory virtual sensor applied to CEM.

The total processing time from image acquisition to trajectory output was approximately seven seconds on a Sun 3 with a floating point coprocessor.

2.5 EXPERIMENTS WITH THE ALV

During the months of July and August, 1987 we successfully demonstrated our obstacle detection and avoidance algorithms on board the Autonomous Land Vehicle at MMAC. The perception system consisted of the CEM and VMT virtual sensor code running on a Sun 3 resident on the ALV producing the maximum safe distance along seven linear trajectories. This information was used by a corresponding planning system to avoid obstacles. A map based planner provided start, subgoal, and goal points while a local planner monitored progress of the vehicle toward each subgoal. All planning modules resided on an off board Lisp machine which communicated with the perception system and vehicle control systems over a radio link.

For these experiments, we chose a site rich with obstacles and potential paths for obstacle avoidance. The area was a hillside containing steep slopes (some over 15 degrees), rock outcrops, large scrub oaks, and small junipers (25 inches high), as well as a narrow sign post. In addition, the area included complex ravines, caused by rain runoff, spanning a range of widths from 6 to 24 inches and depths from 4 to 30 inches. The soil was loose and sandy in the vicinity of the ravines, as well as on portions of the hillside. The ALV can traverse slopes with vehicle pitch and roll of less than 15 degrees making the hill generally navigable except for outstanding terrain features. The remaining surface consisted of grass which was mowed to be less than 9 inches tall, as required by fire safety codes. The difficulties associated with this cross-country terrain were sufficient to require caution by a human driver.

Figure 7 shows a sequence of nine range images and the corresponding CEM and vehicle model trajectories. The ALV was running completely autonomous using a simple on board behavior which chose the longest trajectory biased toward a goal heading. The range images were acquired from the ERIM laser scanner at approximately 8 second intervals with the ALV moving at 0.5 meters per second. The terrain is steeply sloped in places (close to 15 degrees) which causes the vehicle and scanner to tilt forward, shortening the effective visibilty. A large rock outcrop surrounded by trees is visible on the left for the duration of the sequence. The small object visible in the second through fourth frames is a juniper bush about three feet tall, which was adequately avoided by the on board behaviors controlling the vehicle.

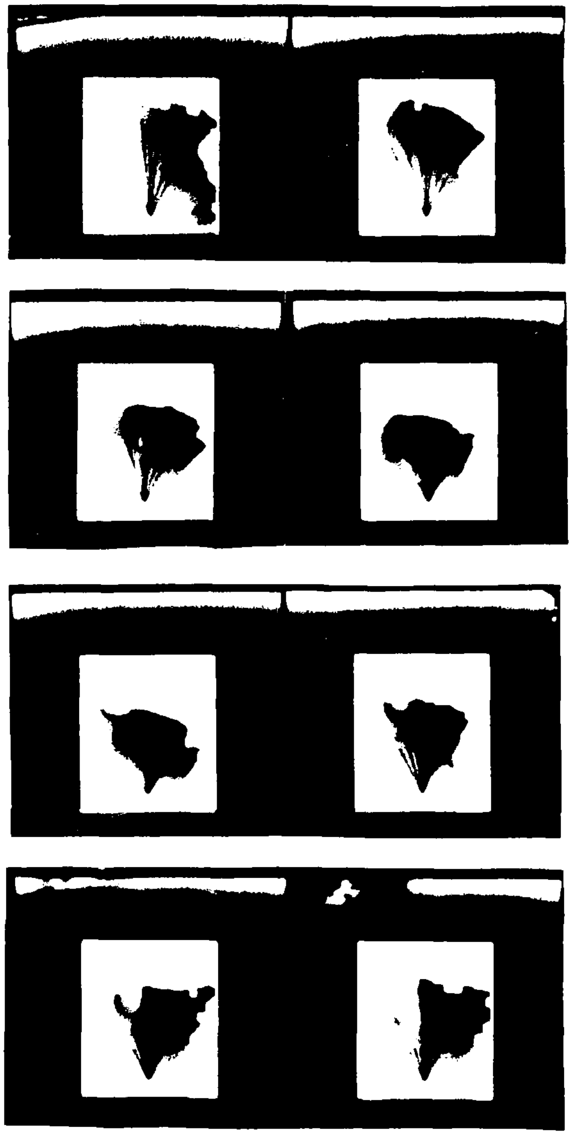


Figure 7. Sequence of range images and CEM's with vehicle model trajectories, in order from left to right and top to bottom. On the left is a large rock outcrop with tall trees, while a small juniper is visible in frames two through four. Vehicle pitch and roll vary from 5 to 13 degrees.

3. ENHANCEMENTS

As described in Section 2, the vision system currently calculates the clear distance along each trajectory (eight bits of information per trajectory). Since seven trajectories are computed, a total of 56 bits of information are used by the planner for each range scan. Though the system works remarkably well, we have noticed several problems due to this paucity of data flow between the perception and planning modules. The planner has no indication of how dangerous a clear trajectory really is. For example, the planner may unknowingly choose fairly steep (i.e. near 15 degrees) paths or trajectories that pass very close to obstacles. These potentially dangerous paths are technically not obstacles, but there may be safer routes available. We have designed a gradient technique for obstacle detection and a downlooking Cartesian map of costs for vehicle traversability called the Cartesian Weight Map (CWM) in order to extract and combine more information from the range data. With this added information, the planner will navigate more intelligently.

3.1 VEHICLE MODEL IMPROVEMENTS

In Section 2.4, we discussed the application of the vehicle model using seven linear trajectories at predefined headings in the CEM. However, the planner controls the vehicle through speed and turn rate commands which result in curved trajectories. Furthermore, subsequent to the calculation of the CEM from a particular range image, the vehicle model test was performed beginning at the vehicle's location when the range image was acquired. Figure 8 shows a CEM with several curved trajectories generated using speed and turn rate and applied at the location of the vehicle when vehicle model processing began. These capabilities are important since eventually we hope to be able to compute the CEM asynchronously at frame rates (one half second for the ERIM scanner) and run the vehicle model on the most current, available CEM. Along these lines, we have developed a virtual sensor which merely checks the current path of the vehicle as fast as possible and returns the safe distance available along that path to the planner. The CEM is calculated asynchronously in a separate process on the Sun and passed to the virtual sensor which constantly checks the current path on the most recent CEM, updating the vehicle position as it moves. In this way, the planner is never moving the vehicle blindly along its current direction for more than a fraction of a second.

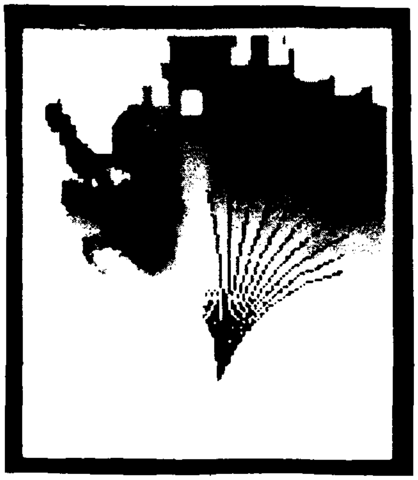


Figure 8. CEM from Figure 3 with curved trajectories.

3.2 GRADIENT OF GAUSSIAN (GOG) OBSTACLE DETECTION

Our current VMT strategy simply tells the planner how far the vehicle may go along a given trajectory. The planner has no idea how close the trajectory passes by an obstacle. In simulation and in real life the vehicle tends to pass too close to obstacles since no part of our current system deals with side clearance. Using a wider vehicle in the vehicle model test would not be a satisfactory solution since this would change the definition of all kinds of obstacles. Also, a wider vehicle model would never allow the actual vehicle to pass through a narrow opening. We could ease this problem by searching additional trajectories on either side of the chosen trajectory. As mentioned above, curved trajectories should be searched. Rather than expand our system to search the large set of possible curved trajectories with the computationally expensive vehicle model, we have developed the Gradient of Gaussian (GoG) obstacle detection technique. GoG returns an approximate tertiary map which marks traversable, nontraversable, and unknown areas. This representation can be used to quickly generate a few trajectories of in.crest, which are then verified by the vehicle model.

GoG obstacle detection in the CEM is analogous to edge detection in video images. Edge detection in images typically requires smoothing and differentiation. The GoG method first smooths the CEM with an appropriately sized Gaussian, and then thresholds the magnitude of the gradient. We choose the threshold value to be 15 degrees, the maximum slope that the ALV can safely traverse in any direction. The standard deviation of the Gaussian must be large enough to smooth out high frequency objects that are not obstacles (i.e. small things like cans) and yet allow major terrain features which are obstacles to remain. We have empirically determined that a standard deviation of three feet yields an obstacle detection performance similar to that of our vehicle model. An application of the GoG technique to the example CEM is shown in Figure 9.

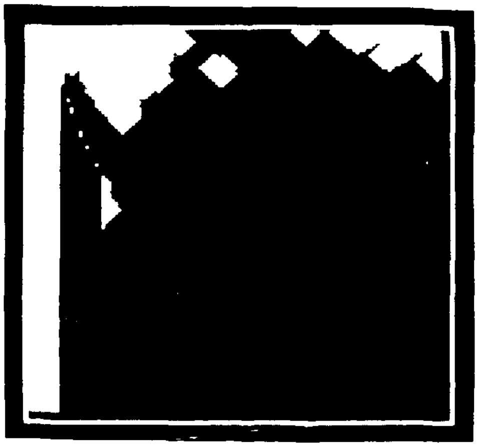


Figure 9. Gradient of Gaussian on CEM from Figure 3.

Since, the GoG technique is a gross simplification of the vehicle model, we require that potential paths and the final chosen path be checked by the vehicle model. Furthermore, edge detection techniques, like GoG, have difficulties with the localization of features. That is, when an image (or CEM) is smoothed, the location of the real edge (or obstacle) is difficult to determine without additional processing. This smearing of obstacles is usually not a problem since the vehicle normally stays far from obstacles. However, when the vehicle must pass through narrow areas where accurate localization is necessary, the constrained space is searched with the vehicle model to find clear pathways.

3.3 CARTESIAN WEIGHT MAP

We originally developed the Cartesian Weight Map (CWM) to prevent the vehicle from passing too close to obstacles. Like the CEM, the CWM is a down-looking map in the local coordinate system of the vehicle. A single pixel value in the CWM contains a weight identifying the cost of traversing the corresponding pixel in the CEM. Obstacles found using GoG are given an infinite weight in the CWM so that no path will ever travel through an obstacle. To solve the problem of the vehicle straying too close to obstacles, all other CWM pixels are given weights which exponentially decay with distance from the nearest obstacle. These weights are set by using a modified medial axis transform (or skeletonization) operation that consists of local propagation among neighboring pixels. A dynamic programming algorithm (Dijkstra's) is used to find the least cost path from the current vehicle position in the CWM to every other pixel in the CWM. We have simulated a suggested path for our example CEM as shown in Figure 1ᑎ We have tested the use of the CWM in a sophisticated simulation environment and found that the CWM technique safely guides the vehicle equidistance between obstacles and avoids small cul de sacs.

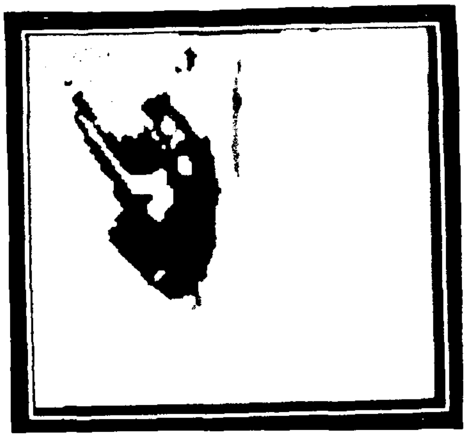


Figure 10. Least cost path on CEM from Figure 3.

So far we have discussed the use of the CWM in terms of a single virtual sensor, nearest obstacle distance. However, the CWM provides a general framework for the combination of various virtual sensors that deal with navigation. For instance, one virtual sensor could penalize some areas in the CWM which are bumpy and reward the regions that are smooth. Perhaps another virtual sensor could update the CWM based upon the slope of the ground. This system of multiple virtual sensors concurrently updating the CWM (shown in Figure 11) is consistent with our hierarchical planning system. To combine slope and nearest obstacle distance into a single weight we have to answer questions such as, "Would you rather take a path that passes within 2 feet of an obstacle or take a path that rolls the vehicle 10 degrees?" The local planner is best able to deal with questions of this type. The local planner may decide to use a combination function that fuses the different types of information into one weight at each pixel. This combiner function would be a nonlinear combination of the outputs from different virtual sensors. The combiner function should not be fixed, but will change with various environments, speeds, and mission objectives. As before, a standard dynamic programming algorithm can be used to find the minimum cost path through the CWM.

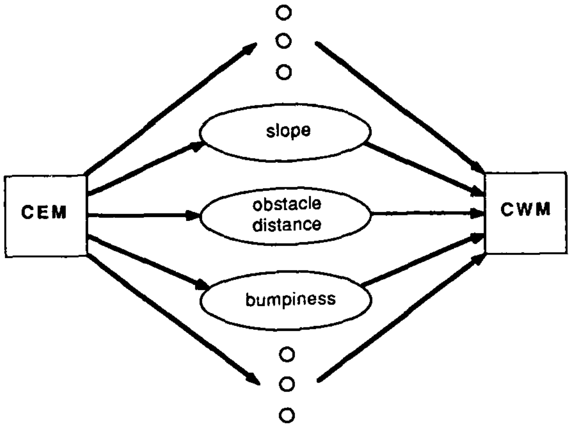


Figure 11. Updating the Cartesian Weight Map.

3.4 PARALLELISM

A great advantage of the CEM and CWM systems is that they are extremely parallel. In cooperation with MIT in March, 1987, we implemented the CEM construction algorithm on the Connection Machine (Hillis 1985). Creating a 128 x 128 (6 inch resolution) CEM from a range image took less than 0.5 seconds on the CM-1. The unoptimized Lisp code was floating point intensive, so we expect a major speed increase when we have access to a CM-2 (the next generation Connection Machine with special floating point chips and other enhancements). During our two week programming spree on the Connection Machine, we had extra time to develop the GoG obstacle detection technique which requires only an additional 8 milliseconds of computing time.

We believe that the virtual sensors for finding distance from obstacles, bumpiness of terrain, and steepness will also operate on the order of milliseconds on the Connection Machine. If each pixel in the CWM is allocated a processor, Dijkstra's algorithm is also extremely fast. Our conservative estimate is that 256 iterations (longest path length) of this algorithm using integer weights will take less than 200 ms on the Connection Machine.

4. CONCLUSIONS AND FUTURE PLANS

We have presented an operational perception system for cross-country navigation which has been verified in both simulated and real world environments. Range image data is transformed into an alternate representation called the Cartesian Elevation Map. Virtual sensors, including those which use a detailed vehicle model, operate on the CEM to produce traversability information along selected trajectories. This information supports a real-time reflexive planning system. Enhancements to the current implementation of the perception system include curved trajectories for the vehicle model and the Gradient of Gaussian obstacle detection and Cartesian Weight Map techniques.

Our immediate plans include further testing during additional experiments on the ALV in November, 1987. We are currently porting our algorithms to the WARP multiprocessor which will be available on the ALV in the near future. We also hope to incorporate the CWM into the perception system, as well as develop new virtual sensors which use range and color imagery. Higher levels in the perception system will also be developed to use the output of virtual sensors for object recognition.

Acknowledgements

We gratefully acknowledge the continued support of the Hughes Artificial Intelligence Center staff, especially Karen Olin, David Tseng, and members of the planning section. We would also like to thank the ALV team at Martin Marietta Aerospace Denver for their excellent support of our efforts in performing the cross-country experiments.

REFERENCES

M.J. Daily, J.G. Harris, and K. Reiser, "Detecting Obstacles in Range Imagery," *Proc. Image Understanding Workshop*, pp. 87-97, February, 1987.

M.J. Daily, J.G. Harris, D. Keirsey, K.E. Olin, D.W. Payton, K. Reiser, J.K. Rosenblatt, D.Y. Tseng, and V. Wong, "Autonomous Cross Country Navigation with the ALV," to appear in *IEEE Robotics and Automation*, Philadelphia, PA, *April, 1988.*

M.J. Daily, J.G. Harris, K.E. Olin, K. Reiser, D.Y.Tseng, and F.M. Vilnrotter, "Knowledge-Based Vision Techniques Annual Technical Report," U.S. Army ETL, Fort Belvoir, VA., October, 1987.

W.E.L. Grimson, *From Images to Surfaces: A Computational Study of the Human Early Visual System*, MIT Press, Cambridge, MA., 1981.

M. Hebert and T. Kanade, "First results on outdoor scene analysis using range data," *Proc. Image Understanding Workshop*, pp. 224-231, December, 1985.

W.D. Hillis, *The Connection Machine*, MIT Press, Cambridge, MA., 1985.

J. Lowrie, R. Greunke, K. Gremban, M. Thomas, B. Gotthard, R. Koenig, G. Celvi, and R. Rehn, "Autonomous Land Vehicle Annual Report," ETL-0413, U.S. Army ETL, Fort Belvoir, VA., December, 1985.

R.E. Sampson, "3D range sensor via phase shift detection," *IEEE Computer* **20**(8), pp. 23-24, August, 1987.

D. Terzopoulos, "Multi-level Computational Processes for Visual Surface Reconstruction," *Computer Vision, Graphics, and Image Processing* **24**, pp. 52-96, 1983.

D.Y. Tseng, M.J. Daily, K.E. Olin, K. Reiser, and F.M. Vilnrotter, "Knowledge-Based Vision Techniques Annual Technical Report," ETL-0431, U.S. Army ETL, Fort Belvoir, VA., March, 1986.

D.M. Zuk and M.L. Delleva, "Three-Dimensional Vision System for the Adaptive Suspension Vehicle," Final Report No. 170400-3-F, ERIM, DARPA 4468, Defense Supply Service - Washington, January, 1983.

Overview of the SRI Cartographic Modeling Environment [*]

Andrew J. Hanson and Lynn H. Quam

Artificial Intelligence Center
Computer and Information Sciences Division
SRI International

Abstract

The SRI Cartographic Modeling Environment has been created to support research on interactive, semiautomated, and automated computer-based cartographic activities. The underlying image manipulation capabilities are provided by the SRI ImagCalc^{TM} system. The cartographic features and data that can be entered include multiple images, camera models, digital terrain elevation data, point, line, and area cartographic features, and a wide assortment of three-dimensional objects. Interactive capabilities include free-hand feature entry, altering features while constraining them to conform to the terrain and lighting geometry, adjustment of feature parameters, and the adjustment of the camera model to display the scene features from arbitrary viewpoints. Cartographic features are depictable either as wire-frame sketches for interactive purposes or as texture-mapped renderings for realistic scene synthesis. High-quality simulated scenes are created by texture-mapping images onto terrain data and adding renderings of cartographic features using depth-buffering and anti-aliasing techniques. Motion sequences can be created by choosing a series of camera models and rendering the simulated appearance of the scene from each viewpoint.

1 Introduction

Among the cartographic applications upon which computer-based techniques can have a substantial impact are the following:

- **Cartographic compilation.** Manual cartographic compilation is expensive and traditional techniques limit the ways in which constraints and previously-compiled information can be incorporated into the process. The task of compilation requires the construction of world-registered three-dimensional models from a body of sensor data and geometric information sources. Such information is now used not only for the creation of paper-based cartographic products, but also for the establishment of digital databases that are used for two-dimensional cartographic depictions and three-dimensional tactical simulations. Without automation, the increasing demands for timely, detailed cartographic information will vastly exceed the resources available.

- **End-user consumption of cartographic products.** A growing number of end-users need to interact directly with digital data, without requiring the use of paper media. These same end-users must be able to select, customize, and update the distributed data to meet rapidly changing situations or unforeseen requirements. These needs have led to the concept of the "Deployable Digital Database," in which digital media replace paper media for information storage and retrieval.

- **Generation of mission-planning and training scenarios.** Mission planning and training can be greatly enhanced by the ability to visualize spatial environments. This can be accomplished within a computer graphics system by generating synthetic static views and motion sequences incorporating detailed cartographic feature models and texture maps of real imagery projected onto the features and terrain.

The SRI Cartographic Modeling Environment has been developed as a sophisticated experimental research tool for exploring three-dimensional modeling tasks. While the system has many capabilities in common with other Computer Aided Design (CAD) systems, it is particularly well-suited for evaluating computer-based approaches to the cartographic applications noted above. It supports a variety of interactive facilities for creating, editing, viewing, and rendering three-dimensional models of world objects, cartographic features, and terrain. The modeling capabilities may be used independently or in conjunction with multiple, metrically calibrated digital images. Interaction with geometric models is characterized by intuitive simplicity and by innovative techniques for exploiting geometric and data-driven constraints in the manipulation process. Synthetic views of a scene may be constructed from arbitrary viewpoints using terrain and feature models in combination with texture maps acquired from aerial imagery. Figure 1 gives a schematic overview of the system's organization and capabilities.

This system extends the two-dimensional capabilities of the ImagCalc^{TM} image manipulation system to three-dimensional space. The geometric parameters of cartographic features and camera models are defined in a three-dimensional world coordinate system. Each image has an associated camera model that defines a ray in space for each pixel in the image. These camera models are used to project world coordinates to image coordinates; when a world coordinate corresponding to a point in the image is required, the camera models also serve to project a ray from the image to the world, so that its intersection with terrain and object models can be found.

Among the potential applications of the system we note the following:

- Studying human interface techniques for enhancing productivity in the computer environment.

[*] This research was supported by the Defense Advanced Research Projects Agency under Contract No MDA903-86-C-0084. Some aspects of this research were also supported by National Science Foundation Grant No. IST-8511751.

- Developing improved specifications for future, more specialized application environments.
- Serving as a flexible background environment in which to develop and test semiautomated and automated cartographic analysis techniques.
- Performing one-of-a-kind tasks such as creating a unique set of models to be used in a simulated image, or making a demonstration videotape of a simulated mission scenario.

The distinguishing features that set the Cartographic Modeling Environment apart from more conventional CAD systems include:

- Registration of multiple data sources, including stereographic or multiple images, terrain elevation models, and three-dimensional object models. to the same world coordinate system. This capability is unique in that it permits object model entry to be driven by *sensor* data such as actual images.
- Use of lighting models, terrain elevation data. and other geometric knowledge to constrain and facilitate data entry. The exploitation of *constraints* in the interactive modeling process potentially increases the efficiency of the human operator.
- Registration of local coordinate systems to UTM's, latitude-longitude, and other cartographic coordinate representations. The use of real-world coordinate systems enables the system to exploit specific world knowledge, e.g., by computing the sun position for a particular location at a particular time of day.

The remainder of this document is organized as follows: Section 2 describes the basic facilities that are available in the system for the interactive user; Section 3 gives an overview of the internal structures of the system that would be used by a programmer to build applications within the environment; Section 4 outlines the hardware and software characteristics of the current implementation. The final section concludes with brief summary.

2 Interacting with the System

The Cartographic Modeling Environment can be used in two very different ways. This section is devoted to a description of the interactive tasks that can be carried out on the system with little or no reference to the programming environment; no significant knowledge of the computer system itself is required. since all operations are carried out by pointing to the graphics display with the mouse and performing simple menu operations with the mouse and/or keyboard. In the subsequent section, we describe the programming interface to the system for those who need to go beyond the facilities provided in the interactive user interface.

In this Section, we first discuss the data entry modes and cartographic display capabilities that are supported. Next. we describe the nature of the user interface. the types of data sources used. and the components of an interactive scene view. Finally. we give an outline of some of the interactive feature manipulation activities possible within the system.

2.1 Data Entry Modes

The system supports three major approaches to the entry of cartographic data:

- **Manual.** A human operator can directly interact with the computer system in many ways, ranging from viewing a library of images and sketching cartographic features to creating synthetic scene views. In this mode, the user typically adjusts the position and parameters of a feature using direct visual feedback from the underlying image data sources.
- **Semiautomated.** The computer has the ability to perform certain operations to enhance the human user's effectiveness substantially while remaining under the human's direct, interactive control. Examples of such operations are the display of registered world points on multiple images, no two of which need be fusible as stereographic pairs, the display of illumination rays intersecting any world point, and running local feature-extraction operators that depend upon being given a reasonable starting cue.
- **Fully Automated.** Very few cartographic compilation procedures can be fully automated using present technology. However, as automated techniques mature, it is apparent that they will need to coexist with the manual and semiautomated compilation methods. Furthermore, even fully automated techniques need an environment in which the human initiator of the analysis process can interact with and select the domains and goals of automated operation. Thus we see our system as a natural framework in which to perform the staged transition from manual to automated computer-based cartographic analysis.

2.2 End-User Data Display Capabilities

Facilities similar to those required by end-users are supported as well. Among the specific viewer-oriented capabilities are:

- **Data Selection.** The user can choose a customized subset of the features available. thereby uncluttering the display to make his task more efficient.
- **View Manipulation.** A data set can be viewed using an *arbitrary camera model. whose position. orientation.* and internal parameters are controllable as those of an aerial observation platform might be.
- **Creation of Synthetic Views.** Once an area of interest. a data set, and a camera view have been selected. the system can create synthetic views to show the operator how the area might look in real life from the chosen viewpoint.

We remark that soft-copy views constructed using such tools have *many potential advantages over hard-copy cartographic* products.

2.3 User Interface Components

The interactive environment consists of the following components:

- **Graphics Screen.** The graphics screen of the workstation is divided into a number of windows. Each window holds a stack of views, and each view consists of a camera model, an optional image corresponding to the camera model, and a set of cartographic feature databases. Wire frame models of the cartographic features are overlaid on the image, if present, using the associated camera model to project from world coordinates to window coordinates.

The features themselves are displayable either in terms of wire frames or texture-mapped renderings. The wire frames are easily moved around in real time and have special sensitive regions used as interactive handles for the mouse functions described in the next paragraph. When realistic but static grey-scale images of the scene are desired instead of real-time manipulation, each feature has an alternative display method that renders texture-mapped faces into a depth buffer with anti-aliasing.

- **Mouse Pointing Device.** A variety of operations are accessible by pressing either the left, middle, or right mouse buttons while some combination of the four state-modifying shift keys (Control, Meta, Super, and Hyper) is held down. Depending on the operation, the movement of the mouse cursor may serve to change a parameter of the object. When an operation involves several steps, a sequence of prompts for mouse and keyboard actions appears on the screen. Two of the mouse operations, the *Create Object* function and the *View Menu*, are specifically for the creation and manipulation of three-dimensional scene structures. The remainder of the operations available directly from the mouse interface, shown in Table 1, are standard ImagCalc operations that deal mainly with the world of two-dimensional screens and digitized image arrays. A superset of the ImagCalc mouse-driven operations can be invoked from an optional pull-down menu bar on the main display screen. These operations are detailed in the ImagCalc manual [Quam, 1988] and will not be described here.

  When the mouse cursor touches a sensitive area of an object's interactive depiction, the menu of operations available to the mouse immediately switches to one appropriate to the object. A typical set of such operations is shown in Table 2 for a simple cube-shaped object.

- **Keyboard.** The keyboard is available for entering text information. Typical operations in the interactive mode would involve typing in parameters on the menu attached to a specific object to modify its current state or configuration, or specifying a file name for loading a data set. Simple Lisp operations typed on the keyboard are interpreted in the Lisp Listener window described below.

- **Lisp Listener.** The Lisp Listener is a window in which keyboard input is passed to the Lisp interpreter and executed. Many mouse operations on windows return the data structure pointed at to the Lisp Listener. In this way, one can assign symbols to individual images and cartographic objects to facilitate their later examination and manipulation.

2.4 Information Sources

Among the types of information that the system is designed to handle are the following:

- **Images.** Digitized photographs or similar sensor data from geographic areas relevant to modeling and feature extraction tasks. Associated with the images should be supplementary data such as geographic coordinates, camera models (or equivalent sensor-to-world transformations), illumination information, exact times that allow reconstruction of the sun position for outdoor scenes, sensor characteristics, sensor response information, and atmospheric properties at the time the images were generated. We note that, at the current time, we are aware of no standard sources of data that include the necessary atmospheric or sensor response information; this lack severely handicaps any efforts that might be made to correct the photometry for atmospheric and film processing effects.

- **Cartographic Products.** Existing cartographic products, either in the form of digital feature data or digitized images of hard-copy products, can be used in the same way as images to provide a world-registered context for the entry or editing of additional cartographic information. Existing low-resolution data can be extremely effective in guiding a high-resolution feature-acquisition task.

- **Digital Terrain Elevation Data.** When available, terrain elevation data should be registered to the corresponding images. This permits the effective use of three-dimensional constraints during feature entry, the construction of accurate synthetic images, and the generation of cartographic products containing elevation information.

- **Compiled Feature Data.** Cartographic feature databases are needed for the generation of cartographic products meeting the needs of tasks for which an image alone is insufficient. Ideally, the system should support the interpretation of precompiled feature databases, the editing of any item in such a database, and the interactive entry of new database features.

- **Feature Models.** In order to support the entry of new features, a library of predefined prototype feature models is required. Such models would be used either by an analyst generating a distributable feature database product, or by an end user updating an existing database. Feature models must support interactive modes that allow them to be easily moved to correct geographic positions and adjusted with respect to their internal parameters.

2.5 View Components

Each view in the stack on a window consists of a fundamental set of components tying together information sources such as those described in section 2.4. The components of a view are the following:

- World-to-view coordinate transformation;

- Cartographic object databases;

- *Optional image*, which must correspond to the coordinate transformation if present;

- Optional terrain elevation model.

The basic operations available on the view include the functions on the *View Menu*, summarized in Table 3. To begin

Shift Keys	Mouse Left	Mouse Middle	Mouse Right
none	Select	Show Selected	Menu
C	% Reposition	Zoom In	Zoom Out
M	Recenter	Copy To Here	Move To Here
MC	Cycle Stack	Kill Top	Kill Stack
S	*X-Slice*	*Y-Slice*	*Histogram*
S C	Set Tandem	Show Tandem	Clear Tandem
SM	Fast Scroll	Fast Zoom In	Fast Zoom Out
SMC	Rev Cycle	Describe	Pop Multiple
H	Window	Scroll Window	Magnify Pane
H C	Print Window	Plot 3-D	Inspect Stack
H M	Scale Rotate	Stretch	Negate
H MC			
HS	Flip-X	Flip-Y	Rotate
HS C	Recenter Others	Create Object	View Menu
HSM			
HSMC	Edit Viewpoint	Push Displayed	

Table 1: The view operations available directly from the mouse when no object is selected. To get access to the functions on each line, the corresponding state-modifying keys must be held down while the mouse button is depressed. The symbols are *C* for Control, *M* for Meta, *S* for Super, and *H* for Hyper.

Shift Keys	Mouse Left	Mouse Middle	Mouse Right
none	Move UV @Z	Move Z	Menu
C	Move UV on DTM	Move W	Move Z
M	XY Sizes	XY Rotation	Z Size
MC	Drop Z	Sun Z	Drop W
S	UV Roll	Reorient	W Rotation
S C	Set Texture	Remove Texture	
SM	Azimuth-Elevation	Z Rotation	Z' Rotation
SMC	Set Face Texture	Remove Face Texture	
H	Clone	Delete	Drop/Obj
H C			
H M	Open Vertices	Close Object	Change Superior
H MC			
HS	Blank	Render	Refresh Views
HSM			
HS C			
HSMC			

Table 2: The functions available directly from the mouse when a cube object is selected. To get access to the functions on each line, the corresponding state-modifying keys must be held down while the mouse button is depressed. The symbols are *C* for Control, *M* for Meta, *S* for Super, and *H* for Hyper.

Database Operations	View Operations
Configure Databases	New View Transform
Add a Database	Render Terrain
Add All Databases	Render Objects
Remove a Database	Refresh Window
Sensitize a Database	
Hide a Database	
Expose a Database	

Table 3: *View Menu.* The model-oriented operations performable on a view.

Flat-Surface Objects	Curve-Based Objects
Box	Flight Path
Building	Open Curve
House	Closed Curve
	Ribbon
Curved-Surface Objects	**Mensuration Objects**
Half Cylinder	Crosshair
Cylinder	Epipolar Crosshair
Superellipse	3D Axes
Superquadric	Ruler
SuperSketch Object	
Utility Objects	**Group Objects**
3D Text	Composite Object
DTM Mesh	Cartographic Database
Camera	
Sun Ray	

Table 4: *Create-Object Menu.* The classes of objects that may be instantiated.

working with a view, one must either create a bare transform from the *View Menu*, or manually load an image file and set up the required items on the resulting viewpoint in the window. Once a view has been established, one can perform such operations as adding, editing, deleting, and cloning objects, copying a viewpoint to a different window without an image, changing the simulated camera position, acquiring texture maps for object faces, and generating simulated scene views.

2.6 Feature Manipulation

Features are instantiated either by cloning existing features or by selecting from a pop-up menu of feature types invoked by *Create Object*. The object types currently available on this menu are summarized in Table 4. Each feature contains a set of subfeatures (such as polyhedral vertices, edges, and faces) that may be sensitive to the mouse pointing device. Whenever the mouse is pointing sufficiently close to a subfeature, it is highlighted to indicate feature selection, and a prompt window on the screen is updated to indicate the name of the feature and the menu of operations available using the left, middle, and right mouse buttons.

A feature's geometric parameters can be dynamically modified using a variety of operations that exploit continuous mouse-to-screen feedback. For example, one of the operations uses the mouse to drag an object around on the surface of the surface of the terrain model associated with the view. As the mouse changes the position of the object in the window, a corresponding ray from the camera is intersected with the terrain model to determine the world position of the object. All visible views containing the feature are continuously updated to reflect changes in the object's parameters.

There are several types of "utility" objects that provide specialized user-interface capabilities. Among these are

- **Camera model objects.** These are graphical features that allow the user to interactively modify the parameters of a camera.
- **Sun ray object.** The sun ray lets the user control the position of the sun in a view. The ray is adjusted and then set to determine the way in which illumination is taken into account in rendering operations.
- **DTM Mesh.** A mesh representing a small patches of the digital terrain model (DTM) can be moved interactively around the scene to help the user visualize the local terrain characteristics.

Views with new camera models can be created by copying existing views and modifying their camera models. Initially, these new views have no associated aerial image, and only the wire frame depicitions of the feature databases are visible. Images can be generated for new camera models by texture mapping an existing image onto the terrain model and rendering the feature databases.

3 Programming Tools

In order to create new scenarios and test the feasibility of new concepts, one must be able to manipulate and augment the internal system data structures. Here we give a brief summary of some of the structures and the associated programming tools and utilities.

The features manipulated by the system are all implemented as instances of *flavors* in an object-oriented programming environment. This means, in particular, that many distinct object types can share or inherit fundamental data fields and *messages*, which are functions that act intelligently within the data structures of a particular object instance.

The fundamental object types used by the Cartographic Modeling Environment, along with a few of the most critical messages they handle, are summarized below:

- **Viewpoint.** The viewpoint, also referred to in this document as a *view*, is a fundamental ImagCalc flavor that holds the information describing each displayable thing on a window's stack. In ImagCalc, the viewpoint contains such structures as digital images, graphs, and curve plots. In the Cartographic system, the Viewpoint relates perspective transformations representing camera models to corresponding images, elevation models, and databases of cartographic objects.
- **Image.** An image is a data structure containing digitized image data. The image formats supported include binary images, 8-bit grey-scale images, integer and floating point images, and multiple-band images such as color images. The image object handles the message `:display-image`

that chooses the best way to display it on the current window, dithering when necessary. Image pixels can be accessed and altered with the :iref, and :iset messages.

- **Perspective Transform.** This family of objects includes both orthographic transforms and 4 × 4 homogeneous perspective transform matrices that relate a world coordinate system to a particular pixel in the window. Film digitization parameters are incorporated when the window pixel corresponds to an actual digital image. By sending the messages :project-to-world and :project-to-view to a transform instance, one can achieve any desired forward or inverse coordinate transformation.

- **Camera-Model Objects.** This object type is the interactive handle by which perspective transform objects (i.e., camera models) can be accessed and modified by the user. It accepts a family of messages supporting the modification of its three-dimensional position and orientation, as well as its focal length and piercing point.

- **Digital Terrain Elevation Model.** Elevation models are arrays of world elevation values in a designated local coordinate system. The model includes a transformation object that translates from the array grid coordinates to the horizontal world coordinates.

- **Two-Dimensional Objects.** These are objects such as text that are strictly related to the display window, and move only in the two-dimensional window space.

- **Three-Dimensional Objects.** This class of objects includes broad families of objects that are displayable and movable in the three-dimensional simulated world. They accept the :draw-on-view message to make a wire-frame depiction, and the :render-on-view message to generate a texture-mapped view of the object. The two major subclasses in this family are *planar-faced* objects, whose faces are true planes, and *smooth-shaded* objects, whose faces are only arbitrarily tesselated representations of a smoothly interpolatable heuristic or mathematical surface.

- **Three-Dimensional Curves.** This family of objects corresponds to roads, boundaries, and delineations of various types. In addition to the usual motion and depiction capabilities, these objects can have individual vertices added, deleted, or edited independently.

Rendering facilities form an important class of the system capabilities. The available rendering operations include

- **Buffered line-drawing.** This allows complex objects to be moved with minimal flicker because the erasure step is buffered to occur at the same time as the next draw.

- **Z-buffering.** Three-dimensional scenes may have arbitrarily complex configurations of objects intersecting each other and the terrain. Depth buffering is provided to handle these situations correctly when rendering a simulated scene.

- **A-buffering.** A substantial improvement in the A-buffer approach to anti-aliasing [Carpenter, 1984] has been developed especially for this system. This facility removes jagged edges and unrealistic artifacts of the uncorrected rendering procedure by computing subpixel contributions of all visible faces affecting the pixel.

- **Terrain Calc Plot.** Texture mapping of an image onto a digital terrain model is handled by the Terrain Calc system. Among the unique techniques incorporated into this system to make the rendering as realistic as possible are the use of a multiresolution image hierarchy and the dynamic selection of local texture map resolutions.

4 Overview of the System Configuration

The current implementation of the Cartographic Modeling Environment is a research-oriented system that is intended primarily to be a tool for feasibility studies, rather than a fully supported software product.

The system is implemented in a combination of Common Lisp and Zetalisp with Flavors on Symbolics Lisp Machines running Symbolics Genera Release 7 system software. Image data and feature files can be stored on any file system on the network that supports a CHAOSNET file transfer protocol. Support is included for Symbolics black-and-white consoles, for the Symbolics CAD frame buffer systems, and for the Symbolics "Hi-Res" system. The system displays grey-scale images on the black-and-white consoles using dithering, and can display 24-bit color images on color systems with only 8 bits of memory using a color dithering technique. Experimental support is also available for the Tektronix SGS-420 stereographic display system, which is used in conjunction with the Symbolics CAD buffer to provide double-buffered 512 × 512 stereo displays that refresh at 60-Hz per eye, 120-Hz overall. The system has not been transferred to other hardware and software configurations at this time; the feasibility of such a conversion effort is expected to depend strongly upon the availability of high-performance graphics systems, the development of correspondingly capable window system standards, and the availability of high-performance object-oriented extensions to Common Lisp.

This article is an initial version of the first of three manuals that are being written to document the system. The documentation planned at this time consists of the following:

- **Overview of the SRI Cartographic Modeling Environment.** A brief overview of the system and its capabilities [this document].

- **Users' Guide to Interactive Use of the SRI Cartographic Modeling Environment.** A description of the tasks a system user can perform interactively, using the mouse and keyboard to issue commands.

- **Programmers' Guide to the SRI Cartographic Modeling Environment.** A detailed description of the system's internal structure for users needing to customize the system to their own needs and build new capabilities within the context of the environment.

In addition, a series of videotapes is planned. An overview presentation will illustrate the basic nature and capabilities of the system and will be a companion to the *Overview* manual; a family of shorter tapes dealing with specific applications within the environment will be produced from time to time.

5 Summary

The current implementation of the SRI Cartographic Modeling Environment has many capabilities that are of interest for evaluating computer-based approaches to digital cartography. In addition, the system is well-suited for interacting with three-dimensional simulated environments and for performing general, computer-aided three-dimensional modeling and rendering tasks. Future development of the system's capabilities will include the following areas:

- Supporting symbolic representations of scene relationships in the style of semantic networks.
- Improving computer-assisted feature entry and constraint exploitation capabilities.
- Extending scene simulation to additional types of sensors.
- Exploring techniques for high-speed manual or semiautomated entry of high-resolution feature data.

Open problems for research in the context of this system include such tasks as simulation of scenes at arbitrary times of day, supporting irregular and topologically complex terrain data, and incorporating the notions of time and physical constraints into the scene-generation facilities to support complex animation requirements.

References

L. Carpenter, "The A-Buffer, an Anti-aliased Hidden Surface Method" Computer Graphics **18**, pp. 103-108 (SIGGRAPH, 1984).

A.J. Hanson, A.P. Pentland, and L.H. Quam, "Design of a Prototype Interactive Cartographic Display and Analysis Environment," Proceedings of the Image Understanding Workshop, pp. 475-482 (February 1987).

L.H. Quam, "ImagCalc Users' Manual," SRI Technical Manual, 1988.

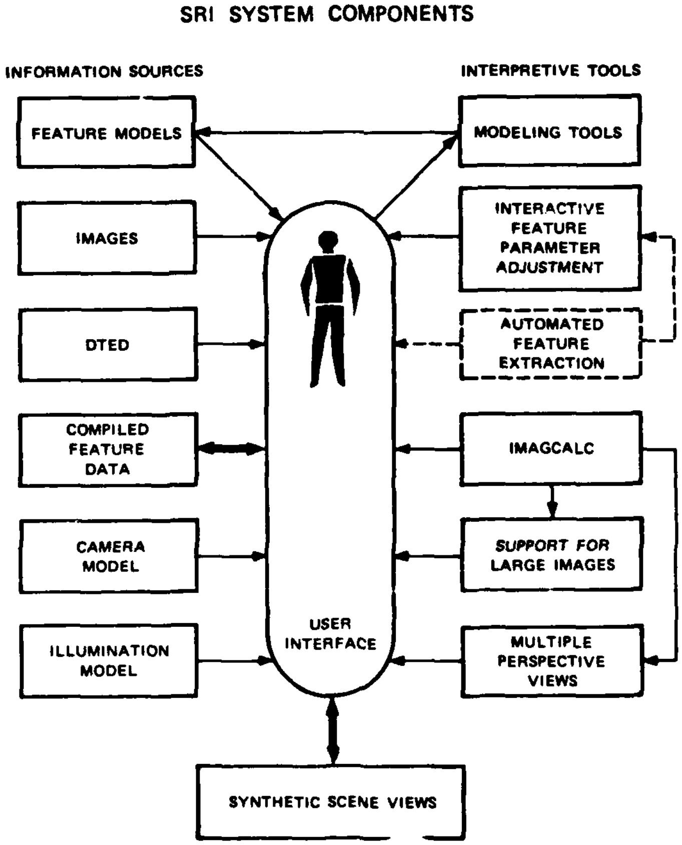


Figure 1: Block diagram of the major components of the SRI Cartographic Modeling Environment.

On The Computational Complexity Of Linear Navigation

John R. Kender[1]
Avraham Leff

Department of Computer Science
Columbia University, New York, NY 10027

0 ABSTRACT

In this paper we formalize a model of topological navigation in one-dimensional spaces, such as single roads, corridors, or transportation routes. We formally define the concepts of a direction and a custom map, as well as some specifications for motor and sensor models, including the idea of sensor synchronicity. We discuss the representations necessary to model and exploit differences between the world itself (here, a version of "Lineland"), the world as perceived by the map-maker, and the world as experienced by the navigator. We demonstrate the difficulty of giving precise meaning to what is meant by a "good" map, and what is meant by a "landmark". We prove that even in the simplest case where both motor and sensor control is error-free, NP-complete problems arise in Lineland, even while attempting to navigate from one single object to another. Forced to use the A* search method, we provide heuristics that appear reasonable for map creation, and give examples of several maps that are "good" under varying criteria.

1 INTRODUCTION

In this paper, we define and explore in a simplified way a model for characterizing a certain class of robotic navigation problems, that of navigation-in-the-large within a linear environment. It is in contrast to much existing work on robotic navigation, which attempts to learn through direct experience a metric map for a small two-dimensional space such as a room [9]. Instead, our emphasis is on the specification, efficient creation, and use of topological maps for a large one-dimensional space, such as a corridor or subway route. The formalization we have chosen we believe will also be useful for other in-the-large problems, whether 1.5-dimensional (such as highway networks), or variations on two-dimensional (such as building floors, forests, or oceans). Since a large body of evidence suggests that humans use topological maps for large-scale navigation [3, 8], the formalization may also prove useful to cognitive scientists.

We believe the value of this paper is three-fold. First, it formalizes several important aspects of the map-maker's relationship to both the world that is abstracted, and to the navigator that will follow the map. Second, it documents the surprisingly large range of problems that navigation even along a line entails. Third, it shows how even in the simplest cases heuristics are not only convenient, but are absolutely necessary, since several of the problems are provably NP-complete; some may be harder still.

The paper is ultimately motivated by two questions: what is a "good" map, and what is a "landmark". In attempting to answer the questions, we investigate classes of maps (and specialize on the concept of "custom" map), and classes of sensor and motor models. Although this paper ignores issues of error, it relates the concept of "good" map to navigational costs of various sorts; they are then demonstrated in with several simple examples.

2 OUR WORLD VIEW: THE WORLD, THE MAP-MAKER, AND THE NAVIGATOR

Many navigation environments are essentially one-dimensional. Doors on a corridor; exits on a highway; bridges or docks on a river; the stops on a train, bus, subway, or even plane route--they all can be abstracted as the (elaborate) values of a function defined on some closed interval of the real line. Because they are static, these environments in some sense are even simpler than the world of "Lineland", an imaginary one-dimensional world postulated by the inhabitants of the imaginary two-dimensional Flatland, in Abbott's prescient novel by the same name [1]. Dynamic one-dimensional worlds do exist, usually when there are multiple navigators; we do not address them.

We find it critical to distinguish three similar but subtly different perceptions of "the world". The one-dimension world as it exists and is experienced by both map-maker and navigator is mathematically rich: it is continuous, it has a distance measure, and objects "embedded" in it can have finite extent. Much of this appears to be extraneous: many such worlds consist largely of the essentially empty space between objects, much navigation can be done by simple object order rather than object distance, and objects can often be considered to be point-like.

We therefore postulate that the map-maker, omniscient and error-free, has abstracted the world into a sequence: that is, the world is conceived as a function over the integers. Empty space, distance, and object extent are ignored. The omniscience of the map-maker ignores the very difficult issues of map induction (see [4]), and the infallibility dodges the issues of partial or errorful information, deliberate camouflage, unexpected events, or even the proper navigator strategies for "believing" a map and for recovering from any errors (strategies that depend, in part on the navigator's model of the map-maker). However, to our map-maker, a plane flight would be simply the sequence, <New York, London, Paris, Rome>.

The map-maker communicates even less of this sequence to the navigator in the form of a "best" "custom map" of "landmarks"; this process is the focus of the paper.

[1] This research was supported in part by the Defence Advanced Research Projects Agency, under contracts DACA76-86-C-0024, and N00039-84-C-0165

Throughout we assume that there is only one navigator, and that the map-maker's omniscience extends to a perfect model of the navigator's sensory endowment: their number, range, responses, and interdependencies. The interesting conflicts that arise when the map-maker mismodels the navigator, their severity, and their recoverability we ignore, but we note that such problems recapitulate many of the issues in user modeling. Similarly, we do not explore how constraining it is for a map-maker to service a diverse clientele of widely varying sensory agents, the sensory endowments of which may not even overlap.

The navigator perceives the world in the most limited way. Sensory range is limited, perhaps only to the current object being experienced, and there is virtually no memory. However, this is an actual advantage; usually the less that needs to be sensed or remembered the better. Although more intelligent navigators can be modeled, this one chooses to ignore most of the world and its features for the sake of efficient traversal. Generalizations to self-teaching, self-correcting navigators (ones that would discover short-cuts while attempting to recover from error, etc.) can certainly be modeled and explored; not here.

Although we do not consider it here directly, it is often the case that the map-maker and the navigator are the same agent; however, for clarity we insist here that they be considered informationally separate, with the only communication between them being the giving of the custom map from the map-maker to the navigator. In human terms, this dual role often occurs when a person reads a road map (that is, consults a representation of the world abstracted in terms of navigator sensory capabilities), summarizes in verbal terms an efficient projected route (that is, creates a "best" "custom map" of "landmarks") , and then follows the road while talking to himself (that is, executes the custom map as navigator). It is important to note the three levels at work here again: there is the world, the abstraction of it, and the custom map.

For consistency, we will refer to the world as "Lineland", the abstraction of it as the "world model" or the "abstract sequence", and the custom map as, simply, the "map". In layman's terms, the world model is usually called, itself, a "map", and the custom map would probably be calleo "the list of directions." However, in some instances, particularly at car rental agencies equipped with electronic custom map-makers, the layman can be given exactly what we call a map, if he has a unique destination in mind. Surprisingly, recent research shows that maps as we define them are preferred by human beings over the Exxon-variety world models [11].

3 FORMALIZATION OF THE NAVIGATOR

We now specify in some detail the definitions and representations that are necessary to investigate the difficulty of navigating in such a domain. In this paper we do not address most issues of error or uncertainty: dynamism in Lineland, partial map-maker information, sensory error, and motor inaccuracies are all ignored. We simply note that there are many wavs to model each, and that the addition of some uncertainties to the problem may even raise its level of difficulty in the complexity hierarchy [5].

3.1 THE ABSTRACTION OF LINELAND

The representation of Lineland as a sequence depends critically on several epistemological assumptions that the map-maker makes with regard to the (single) navigator's sensory abilities and the map-maker's model of them.

To consider objects as point-like requires enough processing intelligence in the navigator to recognize an object as a single object, and to capture, hold, and dismiss the single object from its sensory array; in effect, the navigator can "debounce". This is an important consideration, since the "experience" of an object must be a unified whole, even if the prior and succeeding experiences are identical. Three identical doors on a corridor must give rise to the experience of "door" exactly three times, even though the doors are sensed at some distance, have spatial extent, and are still sensible after some distance, often even when the next door is also sensible. In particular, the experiences of the door's height, color, shape, size, and other features must occur exactly in synchrony, all switching the navigator's appropriate sensors on and off at the same "time", although the navigator may selectively "attend" to any subset of the full sensory array. It is not clear what sort of world model results if these constraints are relaxed.

3.2 THE MODEL OF THE NAVIGATOR'S SENSORS

The map-maker must model the extent to which the navigator can sense remotely; this may be sensor dependent. For example, the sensor might return distant information with less reliability, or the sensor might be distance- or instance-limited. These interesting problems are simplified here by the assumption that none of the navigator's sensors are far-sighted; they can verify the imminent presence of an object only. For the sake of argument, the navigator can be considered to sense by touch only, although there may be many attributes (many sensors) to describe the sensations. Alternatively, the navigator can be considered to be near-sighted, or operating in a fog.

Given the model of unified, immediate perception, Lineland can now be modeled by the map-maker as being a vector-valued sequence; each object in Lineland is reduced to exactly one vector of sensations as experienced by the navigator, one component in the vector for each sensor. We place no restrictions on modalities of the sensors, and they may be any measurable quality such as color, distance, area, texture, shape, number of holes, etc.: an object can be perceived as "green, far, large, rough, square, two". However, we require that each such sensor indicate stimulation by an object by presenting to the navigator a symbol from a discrete set; we bypass the problems of continuous values, multi-dimensional or structured values ("icons"), and all their extensions to probabilistic sensations (for example, "mainly blue", "possibly a face", etc.).

Thus, the map-maker's view of the Lineland is really the navigator's, except omniscient and error-free. Although the map-maker may have at disposal many ways of perceiving Lineland as an occasional navigator, too, only those that the specific navigator can respond to are considered in map-making. Thus the map-maker's view of Lineland for purposes of map-making may vary greatly from any "personal" experience of it. Human examples include the invisibility of IR, UV, SAR, and other non-natural sensations, which nevertheless can be used in directing a non-natural navigator.

The navigator is modeled as having exactly S sensors, and each sensor s_k takes on its values from its associated discrete domain d_k. For example, d_1 might be {red, green, blue}. Thus, formally, Lineland is modeled by the map-maker as <object$_1$, object$_2$, ..., object$_L$>, where each object is an element of the cartesian product of the sensor domains. Thus, <<big,red>, <little,blue>, <big, blue>> is a trivial Lineland model, as is <<New York>, <London>, <Paris>, <Rome>>. The Lineland model sequence inherits a natural ordering;

sometimes we will refer to the predecessor and successor of an element as "left" and "right", respectively.

Although we will never use it, we can also define the model of intervening empty space as an s-vector of "nulls", indicating nothing can be perceived on any sensor other than its normal background. The sensation vector is therefore either all nulls or all non-nulls, and no partially perceived objects are permitted here, although a "partially null sensation" may well be taken as a working definition of camouflage.

Although the map-maker's experience of Lineland is error-free, the navigator's may not be. There are many ways to specify sensor error, with one of the simpliest being a confusion matrix For each sensor, the matrix indicates the probability that a given value of its domain will be perceived mistakenly as another value. Usually the matrix can be extended to include nulls, thus recording the probability of falsely positive and falsely negative existences of objects. Given a navigator that has a large sensory endowment, some representation must be made of sensor-sensor error interaction, which itself may be probabilistic. For example, if total agreement among all sensors is necessary for the establishment of object existence, then as the number of sensors increases the number of falsely negative objects increase, and the number of falsely positive objects decrease.

More irritating is the non-zero probability of the navigator navigating wildly: circling hopelessly under confused sensory perceptions, or charging off beyond any objects and towards either "end of the world". We note in passing that humans are not immune to this behavior. Special considerations then have to be made about error recovery, and even of the semantics of the "end of Lineland". For these and other reasons we assume here that the navigator's sensors are perfect.

3.3 THE MODEL OF SENSOR SYNCHRONICITY

The map-maker must also model the dependencies of the various sensors on each other, since the creation of a map implicitly will attempt to exploit those sensors that are most available and least costly. In fact, map-making can be defined as the selection of the proper subset and sequencing of sensors so as to minimize the navigator's cost, measured in various ways, of proceeding through and attending to the world while attaining a goal location.

Since the definition of a sensor is broad, often it does not correspond to a particular piece of hardware. For example, area and perimeter measures are considered to be separate sensors, although typically they can be calculated simultaneously from common information. More to the point, they can be assumed to be available for parallel sensing, with a cost (if measured in time) equal to the maximum of their individual costs; further, they may imply a further time cost of any prerequisite "low-level" sensing. Thus, whether sensors are "virtual" (as above) or real, they may interrelate in complex ways, having various data dependencies; these dependencies in turn interact with the navigators' ability to simultaneously "attend" to appropriate subsets of them in parallel.

We bypass the difficult question of finding a formalism for modeling these synchronicity relationships in general. Probably some directed graph structure will suffice for data dependency (i.e., (a,b) if sensor a is a prerequisite for sensor b), and probably a separate undirected graph structure will suffice for parallelizability (i.e., (x,y) if sensor x can be run co-parallel with sensor y); the nodes themselves can be labelled with their time or other costs. We simplify our task here by assuming that all sensors are data-independent--they do not depend on the results of any other sensor as a prerequite--and execute in unit time cost and unit "sensor" cost (roughly, a measure of perceptive difficulty). This leaves us to explore the parallelization relations.

The map-maker has the difficult chores of properly selecting a sensor subset and exploiting any sensor parallelism. For example, depending of the surrounding objects with which it appears, the green-far-large-rough-square-two object above might be most efficiently sensed simply by 1) "green", if all the other objects nearby are red, by 2) "green-large", if all the other objects nearby are red or small or both, and the color and area sensors are parallelizable, by 3) "green then large", if all the other objects nearby are red or small or both but the color and area sensors not parallelizable, and the map-maker has noticed that more of the objects are red than small, thus making the color sensor less likely to require the area sensor as backup, or by 4) "large-rough then green", if the situation is as in 3), except that objects tend to be small and smooth, and size and texture are parallelizable with each other, but one or the other or both are not parallelizable with color. Many others are possible, but even counting their number is highly complex, since it involves sequences, subsets, and partitions (all of exponential complexity or greater), under arbitrary side conditions. In summary, part of the map-maker's job is to find the most efficient sequence of disjoint parallel sensor subsets that the environment and the parallelization graph permit.

(An aside concerning linguistics. Sensor sequencing is roughly analogous to the ordering of adjectives in English. For example, the meaning of "a little brown house" is somewhat different from that of "a brown little house", and implicitly communicates some of the nature of the environment. Stretching the point, one occasionally sees written "a brown, little-but-sturdy house", and the like. Further, the analogy also carries over in the sense that there are "data dependencies" governing the sequence of adverbs, adjectives, and nouns in noun phrases: the article always comes first, number comes next, adverbs are immediately before their adjectives, etc.: "the three very brown, little-but-sturdy houses" has only one other allowable sequencing [6].)

Rather than allow the navigator to have an arbitrary parallelization graph, or even one that is partitionable into components consisting of mutually parallelizable sensor cliques, in this paper we investigate two common extremes.

The first synchronicity model we call the purely parallel sensor model ("parallel sensors", for short), which corresponds to the case of the parallelization graph being complete. Thus, all sensors are independent, have unit costs, and are fully co-parallel. There is no sequencing; or rather, the map-maker needs only to select the proper subset of sensors to be run in parallel in the first and only subset of descriptions. In short, instead of a sequence of disjoint parallel sensor subsets, there is a single subset. In this case, the time cost of any sensing is a unit, although the sensor cost is equal to the cardinality of the subset. It is to be noted that the parallel sensor case always requires data independence of sensors, whether or not we have assumed it.

The second synchronicity model is the opposite, and we call it the purely sequential sensor model ("sequential sensor", for short), which corresponds to the degenerate case of an empty parallelization graph. Thus, all sensors are independent, have unit costs, and are mutually antagonistic with regard to parallelism. There are no subsets; or rather, the map-maker considers each sensor to be a singleton set, and needs only to select the proper sequence. In short, instead of a sequence of disjoint parallel sensor subsets, there is a simple sequence. In this case, the time cost of any sensing is greater than one, and less than or equal to the length of the sequence, and the sensor cost equals the time cost. It is to be noted that in the sequential sensor case, the map-maker benefits greatly from any additional data dependencies of sensors, since they impose additional constraints on the allowable sensor sequences; as stated before, we do not assume any.

3.4 THE MODEL OF MOTOR ABILITIES

In general, the map-maker needs a model of the accuracy of the navigator's abilities, especially if the custom map will make use of concepts dependent on distance. For example, a motor model can be a mapping that has as its domain a position, a heading, and a desired distance, and gives as its range a probability distribution of resulting positions and headings. Clearly such a model can be simplified in many ways; for example, the motor performance is often independent of initial position. In Lineland a navigator has only the choice of two headings, "right" or "left", or "+" or "-"; and, under the map-maker's world model of Lineland as a sequence, all positions are integers.

However, in this paper, we will disallow any direct references to distances (we are interested in topological navigation), so the motor model becomes trivial and error-free. The map-maker assumes that the navigator can tell left from right with perfect accuracy, and proceeds from one object to its successor or predecessor in the sequence with perfect accuracy, too. As a side issue, we note that this eliminates any need for the "reassurance" directives that are common in human map-making, and which serve to recalibrate the navigator; e.g. "if you see an X you have gone too far, turn around and look for the Y". Further, this eliminates any need for "universal" directives, which are less often found but make the navigator self-correcting; e.g. "go until you hit the river, and follow it downstream to the Z".

A second component of the motor model is one that is useful in defining a measure of motor work; all things being equal we would prefer a custom map that was less costly in terms of motor effort. Many things can be accumulated towards motor cost; for example, instead of minimizing distance traveled in Lineland, we may choose to minimize acceleration or the number of reversals of direction. We simplify our task by assuming that not only is the motor perfectly accurate, it is perfectly efficient and accumulates no cost.

Thus, in exploring the topology of navigation, we ignore even the metric information that accumulates as total distance traveled. Our concern is the complexity of the custom map and the sensory sequencing, not distance traveled. Thus, like some humans, money is no object, but minimizing cognitive strain is. Our custom maps will emphasize compactness and landmarks, rather than motor energy: "short", "easy", but not "fast". (In this regard, they are like the scripts of natural language processing, which are mostly concerned with the temporal topology of significant events; scripts abstract away all metric time information.) In any case, we note that in Lineland the most direct route is always, trivially, a straight line from start to goal.

4 FORMALIZATION OF THE CUSTOM MAP

We now specify the grammar of the custom map, and the assumptions made between the map-maker and the navigator in its creation. A custom map is simply a sequence of directions for the navigator to follow, and is devoid of any spatial representations. We do not formalize or consider the use of graphic information in communicating a custom map; for example, we do not address the equivalent of coloring in the route on an existing road map, or augmenting a custom map with a sketch commonly "not to scale".

4.1 PARTIAL OBJECT DESCRIPTIONS

At the heart of the map-making process is the selection by the map-maker of certain objects to be sought by the navigator, and the sensors that are necessary to unambiguously identify them. Given a sensor synchronicity model, each of these objects will be described by a sequence of disjoint parallel sensor subsets, and the values that each of these sensors will observe when the object is obtained. Assuming i represents represents the index of the ith sensor and v represents the value to be observed, a Partial Object Description (POD) is defined to be a sequence of sets of ordered pairs, $<\{(i_1,v_1), (i_2,v_2), ..., (i_k,v_k)\}, \{(i_{k+1},v_{k+1}), ..., (i_U,v_U)\}, ...>$.

Since we will investigate only two special cases of synchronicity, the notation can be simplified for our purposes. In the parallel sensor case, a POD is considered to be a set of the form $\{(i_1,v_1), (i_2,v_2), ..., (i_U,v_U)\}$, and in the sequential sensor case, a POD is considered to be a sequence of the form $<(i_1,v_1), (i_2,v_2), ..., (i_U,v_U)>$; in both cases we have dropped a different layer of structuring, so the forms are nearly identical. Further notational simplicity is achieved when the (discrete) sensory domains are composed of disjoint symbol sets. In that case, the sensor index is implicit and the form of the POD is either $\{v_1, v_2, ..., v_U\}$, or $<v_1, v_2, ..., v_U>$, respectively. For example, in English the adjectives describing color, area, and texture are all disjoint, perhaps for this very reason: it is clear that {rough, red, big} implicitly refers to texture, color, and area, respectively, whereas {super, great, fantastic} conveys only a sense of extremity.

4.2 UNIT DIRECTIONS

We define a custom map unit direction (or simply, a direction), to be the shortest command that can effect the navigator's movement. Under our assumptions, it is comprised of two components: a heading ("-" or "+"), and a partial object description (POD) that describes the object to be sought.

(Strictly speaking the heading may be redundant. In Lineland, there is never any advantage for a navigator to proceed away from the goal, although it is possible that he may overshoot it arbitrarily often. If the navigator can keep accurate positional information, such as via the "I am here" variable discussed below, the proper heading towards the goal can always be inferred.)

Thus, in the parallel sensor case, an example would be +{big, red, house}; it means that the navigator should proceed in ascending sequence order until the size, color, and structure sensors simultaneously obtain the mentioned values. In the sequential case, it is nearly identical: -<blue, small> instructs the navigator to travel to the left until a blue object is detected, and if it is also then found to be small, the desired object has been attained.

Clearly, more sophisticated unit directions are possible. Instead of a POD, the map-maker can specify a distance or other metric quantity, such as time, energy, or object count. PODs can be replaced with "compound objects", defined under standard regular grammars. That is, a compound object (CO) can be defined as CO ::= not CO | CO-CO | CO/CO | CO^n | POD, where the semantics are "anything other than this description" ("not a red house"), "the spatial concatenation of two descriptions" ("a red house followed by a big tree"), "the spatial alternation of two descriptions" ("a red house or a big tree"), "n occurrences of the descriptions" ("seven red houses in a row"), or a simple POD. Like all grammars, nesting is possible, generating exotic compound objects such as "three occurrences of big houses next to things other than a big tree or a small pole", which have one noticeable advantage: they are very unlikely to be falsely perceived by mistake.

Since grammars place some constraints on the processing power of the navigator's equivalent of a short term memory stack, we ignore them here. We note however that formal language theory, and in particular the study of LR(k) parsers, addresses some of these issues, since basically a direction based on a compound object is a simple program.

4.3 CUSTOM MAPS

A custom map is now defined as <start, direction^D, goal>, where start and goal are Lineland sequence indices (navigator start and goal positions, respectively), and D is the "length" of the custom map. There are some subtle issues involved, however.

The start represents the starting position of the navigator to the omniscient and error-free map-maker. For now, we ignore the problem of how such a location is determined, by whom, and in what manner. For example, the map-maker may be able to construct a custom map for the navigator, the sole purpose of which is to take the navigator from any unknown position (and/or orientation) in Lineland to a single known location (a "landmark"); this is sometimes called the "parachutist's problem". Or, the navigator can motor in a given direction reporting to the map-maker what is (errorlessly) observed until the map-maker knows the navigator's position (and/or orientation) unambiguously; this is called the "shortest containing substring" problem and a solution can be found in [2]. Or, the twr problems can be combined, with the navigator alternately seeking and reporting. Undoubtably other means exist, including keeping a running "where am I" data structure, or asking other navigators (who may be lost or deceitful, etc.), "Where am I?".

Likewise, we are ignoring the epistemological problems of defining the goal. There is little problem if the goal is imposed by the map-maker; what is more difficult is how a navigator would communicate a goal to the map-maker. If it is by index, the question arises as to where the integer came from ("other knowledge", or curiosity, perhaps). If it is by description, the description would usually have to be unambiguous (unless the navigator is seeking an object class), implying that the navigator has some model of the navigation process and even some knowledge of Lineland in general. There is a stage twist at work here. If the navigator is primitive, his descriptions to the map-maker of where he wants to go are primitive, too, and probably underdetermined ("it's red"); as with the analogous human situation, progress is uncertain just when it is most needed. However, if the navigator is savvy enough to uniquely specify the goal, he may not need the map-maker at all ("three trees east of the big blue circular house").

5 NAVIGATING WITH A CUSTOM MAP

Having defined what a custom map is, we now examine its use. Then we will address its creation, which is one of the central concerns of this paper.

5.1 MATCHING

The most primitive decision a navigator must make (and makes repeatedly) is whether or not its observation of the world is compatible with some partial object description given by the map-maker. We model such an operation by a boolean-valued function "match", which takes as input a POD, and uses the POD to schedule the appropriate sensory information to make the judgment.

In the parallel sensor case, all sensors must match the values in the POD for the value of match to be true; otherwise match returns false. In either case, match also returns a time cost of one unit and a sensor cost, N, equal to the cardinality, U, of the POD subset. Since in this paper we are ignoring metric properties of time and space, we will retain only the sensor cost for our future discussions about map efficiency.

In the sequential sensor case, all sensors must again match the values in the POD for the value of "match" to be true; but as soon as "match" detects a mismatching sensor reading it returns with false (that is, it behaves like the conditional-and construct in programming languages). "Match" also returns a time and sensor cost, N, which are both equal to the number of attempted matches; this is at least one and no more than the cardinality, U, of the POD subsequence. Again, we retain only the sensor cost in future discussions of efficiency.

5.2 TRAVERSAL

We now define the semantics of a successful custom map traversal. The navigator receives a custom map, (start, direction^D, goal), and (re)sets his internal "I am here" integer to the value of start. He follows the k directions, updating the "I am here" value every time an object is encountered, and taking a new direction every time "match" returns true. At the final step, he compares "I am here" to goal; the success of the traversal is defined to be the value of this equality. This last step is critical. It is insufficient to merely follow the directions; they must have the desired effect. Since the map-maker is omniscient and benign, and the navigator has perfect sensors, there is no need to define what happens when the navigator runs off the "end of the world" while seeking a bogus POD.

6 SUMMARY OF ASSUMPTIONS AND EXAMPLE

Having established nearly all the definitions that are necessary, we review the assumptions we have made, and exercise our vocabulary in a simple example.

The world is one-dimensional and static, with obvious objects that can be represented as points. It is distinguished from the sequence of discrete-valued s-vectors that the omniscient map-maker has abstracted it into, and from the custom map that the primitive navigator receives. There are no sensory or motor errors of any kind, and sensors have effectively a range of zero. Sensors are independent of each other, and can be fired in a purely parallel or purely sequential manner. We ignore any motor cost, concentrating on issues of topology. The custom map is made from simple directions, consisting of only a heading and a POD. The navigator and map-maker somehow are aware of the actual start and the desired goal, and the navigator has effective procedures for determining if a navigation has been successful.

Now consider the following world model of a Lineland, which is of length 10 and in which objects are described by a single feature (intuitively, color) taken from the domain {r, g, b}: the sequence is <b,g,r,r,r,b,b,b,g,r>. Since there is only one sensor, the parallel and sequential sensor models are equivalent and we dispense with as much syntactic overhead in the PODs as possible. The custom map maker can therefore give directions of the simple form: +r, or -b.

The navigator asks for a custom map from 2 to 8. The custom map maker replies: (2, +g-b, 8). The navigator traverses the map by (re)setting the "I am here" integer to 2, repeatedly increments it while attempting to attain a match to the sensation of g, and first attains a true match when "I am here" equals 9. Reversing direction, a match to b occurs when "I am here" equals 8. The directions being exhausted, the traversal is successful because "I am here" equals goal.

Note that the custom map-maker could also have said (2, +b+b+b, 8), which is a longer sequence of directions but stiii correct. Or even (2, +r+r+g-b, 8), which is longer still, correct, but in some sense redundant.

7 OPTIMAL UNIT DIRECTIONS

We would like to make the idea of "efficient" and "best" map precise. Unfortunately, we first need some additional definitions and representations to investigate how difficult these questions are.

7.1 TRANSITION AND COST GRAPHS

We introduce two abstract data types, the transition graph and the cost graph. They are abstract data types because, depending on the sensor model and the map criteria being optimized (and, in some cases, on the contents of the world model itself), they are realized as various data structures. Thus, in some cases there are best implemented as arrays, in others as lists. In some cases they are best established before the optimizing map-making algorithm, in others they are incrementally established cooperatively with it, and in still others they are never established at all, but are directly incorporated into the algorithm itself.

Intuitively, the graphs record information about the most efficient unit direction, if any, that takes a navigator from any given sequence index in the Lineland world model to any other index. If there is such a unit direction, the transition graph records it, and the cost graph records its cost (which can be defined in multiple ways). Both graphs conceptually contain a special sentinel wherever a unit transition is impossible; in particular, there is no unit direction that can cause the navigator to remain in the same place.

The problem of attaining an efficient custom map from a start to a goal can then be broken down (conceptually, at least) into the establishment of efficient unit directions, followed by the traversal of the unit direction graph from start to goal. There are times, of course, when this is grossly inefficient, such as when the navigator is traveling a very short distance and a type of search is more appropriate; we do not thoroughly analyze such tradeoffs here.

Intuitively, the graphs are filled in in two stages.

7.1.1 FIRST STAGE; DEFINITION OF "LANDMARK"

In the first stage the transition graph is filled with tentative unit directions of the most inefficient kind, namely the full sensory description of the unit direction's goal. For example, if the lineland world model is <<r,x>, <g,x>, <r,y>>, then the unit transition connecting indices 1 and 3 would initially be given as +<r,y> (under the sequential model, assuming disjoint sensor domains), even though either +r or +y would suffice. This first stage is straightforward and easy: starting at each position j, the abstract model sequence is scanned backwards, and as long as no object matching the description of object$_j$ is found, the unit directive given by (loosely speaking) "+<object$_i$'s full description>" is entered at transition[i,j]. More accurately, what is entered at transition[i,j] is +{<1,v$_1$>, <2,v$_2$>, ..., <S,v$_S$>} under the parallel model, and +<<1,v$_1$>, <2,v$_2$>, ..., <S,v$_S$>> under the sequential model, where v$_k$ is the value in slot k of object$_i$. Lineland is scanned similarly in the opposite direction for directions beginning with the opposite heading.

As an exam^p^le, if the Lineland model is again <b,g,r,r,r,b,b,b,g,r>, then the transition graph looks like the below, where empty entries are considered to be filled with sentinel values.

	b	g	r	r	r	b	b	b	g	r
b		+g	+r			+b				
g	-b		+r			+b			+g	
r	-b	-g		+r		+b			+g	
r	-b	-g	-r		+r	+b			+g	
r	-b	-g		-r		+b			+g	+r
b	-b	-g			-r		+b		+g	+r
b		-g			-r	-b		+b	+g	+r
b		-g			-r		-b		+g	+r
g		-g			-r			-b		+r
r					-r			-b	-g	

In some special cases, there may even be more efficient ways of structuring the abstract transition graph. For example, in the above case the graph is quite sparse; since no row can have more than six entries (two headings times three object descriptions), the graph can be compressed using various techniques. In general, if navigation is in-the-large, by definition there will be a large number of objects with the same object descriptions, and the transition graph will always be sparse and compressible. In fact, the transition graph at this stage of processing can be considered purely binary, since both the heading and the object description are implicit in the structure of the graph and can be fully recovered from the coordinates of any entry that is not a sentinel.

We note in passing that one definition of a "landmark" would be an object whose column in the transition graph is very nearly filled, that is, an object that can be obtained under a unit direction from very nearly everywhere. In the above example, either G object is a fairly good landmark. One can easily define a concept such as the "landmark radius" of an object in the obvious way. Unique objects would then have landmark radii of at least half of Lineland. One can navigate great distances without even knowing their indices; one just remembers their (nearly) unique description.

Because landmarks can be used to navigate over great numbers of objects, they will tend to appear regularly in custom maps. Note that landmarks need not be "reversible"; for example, the fifth object in the above example is the third of three R's in a row, and is only valuable as a landmark while navigating left. This is often the cause of distress in human man-making; in general custom maps are not reversible. The first instance of any object in a long row of like objects is often valuable as a unit direction, but it becomes the very last object in the row when considered from the opposite heading and is nearly useless.

7.1.2 SECOND STAGE

In the second stage of transition and cost graph construction, the transition graph is optimized under the appropriate sensor model, and the cost of the optimal unit direction is recorded in the cost graph; sentinel values in the transition graph giving rise to infinities in the cost graph. In the example above, since there is only one sensor, there is no need to optimize, and the cost graph is trivially derived from the transitions.

But in general, the second stage is more complex, conceptually and computationally, for several reasons. First, there may be more than one optimal sensor subset or subsequence that attains the optimal cost (as in the example of the +r vs. +y above); depending on the overall model of map goodness they may all have to be recorded. (However, in none of our cases is this necessary.) Secondly, the optimization of the entire graph may probably be done in a more efficient way than by optimizing each entry separately; this remains to be demonstrated. But thirc' and most importantly, under many sensor synchronicity models the optimization of even a single unit transition is NP-complete in the number of sensors. We prove it to be so for the cases of purely parallel and purely serial sensing.

7.2 THE COMPLEXITY OF PARALLEL SENSING

We begin with an example. Suppose that the navigator is equipped with three sensors (S = 3), and a portion of the transition graph has determined that the unit direction consisting of "+<object$_j$'s full description>" takes object$_i$ to object$_j$. For simplicity, we assume that each of the sensors is binary valued, and that object$_j$'s full description as a 3-vector is <1,1,1>. (Alternatively, the binary 1 can be thought of as indicating that the given sensor slot has successfully matched its value, whatever it may be; conversely, a binary 0 indicates value failure in the slot.)

Consider the following subsequence of the Lineland model, which indicates the 3-vectors of sensed values for the seven objects from object$_i$ through object$_j$, inclusive; the value "X" represents either a 0 or a 1.

	i							j
s1	X	0	0	0	0	1	1	1
s2	X	0	1	0	1	0	1	1
s3	X	1	0	1	0	1	0	1

Under the syntax of directions assuming fully parallel sensor synchrony, the proper unit direction at transition[i,j] = +{(1,1), (2,1), (3,1)}; by our sensor cost function, the amount of sensor work is S|j-i|, in this case, 3*7 = 21. The question arises if any subset of these three sensors is also a valid unit direction at less cost (clearly, at cost of either 14 or 7).

Since S is relatively small, it is not hard to examine the eight subsets exhaustively; most are trivially inadequate. It is apparent that +{(2,1), (3,1)} is a valid unit direction at cost 2*7 = 14, and that it is the only cheaper valid unit direction. As a general rule, any sensor subset is subject to the constraint that, for each of the interior objects in the range i to j exclusive, there must be at least one sensor that provides a value of 0.

What is valuable about the example is that it is the minimum counterexample to any anticipation that a greedy algorithm would be useful in selecting a subset. A greedy algorithm would build up the sensor subset incrementally by the following method. From those sensors which remain, choose whatever sensor that contributes most to detecting that interior objects are not object$_j$; or, in looser words, choose whatever sensor that provides the most "new" zeros. Following the greedy algorithm in the above example, sensors would be chosen in the order s_1 (which detects that the first four objects after object$_i$ cannot be object$_j$), followed by both s_2 and s_3 in either order, since either of them properly detects that one or the other of the remaining two intermediate objects cannot be object$_j$. Thus, the greedy algorithm would not improve on the initial unit direction.

One is gradually led to suspect that no simple algorithm for selecting the sensor subset would do better than exhaustive search. This suspicion is based, in part, on the observation that the utility of a given sensor depends upon each and every of the other sensors selected before it, in a non-trivial way. A simple count of the number of mismatches (zeros) in a sensor does not indicate how many of those zeros are that sensor's unique contributions to the growing collections of mismatches.

The problem, in fact, is NP-complete. We will show that it is a straight-forward instance of a known NP-complete problem, the "minimum cover" cover problem [5], under a simple one-to-one transformation. The minimum cover problem is one of the 21 "original" NP-complete problems presented in Karp's landmaιk 1972 paper [7].

The minimum cover problem is stated as follows: Given a collection C of subsets of a finite set T, and a positive integer M <= |C|, does C contain a cover for T of size M or less? That is, is there a subcollection C' contained in the collection C, with |C'| <= M, and such that every element of T belongs to at least one member of C'?

All that is necessary is to identify T, C, and M. Take T to be the set of interior objects, that is, object$_{i+1}$ to object$_{j-1}$ inclusive. Each sensor s_k contributes one member subset to the collection C, consisting exactly of those objects that it can verify as not being object$_j$ (i.e. all its "zero" objects). Take M to be any number less than S, the number of sensors. Then the parallel sensor problem becomes the minimum cover problem, since we seek a subcollection of M sensors such that every interior object is verifiable by at least one sensor as not being object$_j$.

Of the additional results that are known about the minimum cover problem, a few are relevant here. The problem is known to be NP-complete even if (translating to sensor terminology) each sensor rejects three or fewer interior objects. In fact, the problem becomes tractable only if each sensor rejects no more than two interior objects, for any interior. Although in some cases such sensors may be built (for example, sensor k may be designed to reject object$_k$ and object$_{k+1}$), this would require the map-maker to identify a virtually unique feature for each object. While not quite as extreme as giving all objects house numbers, it would still require the number of sensors to grow linearly with the number of objects.

Having shown the optimization of unit directions under parallel sensing to be NP-complete, what devices are there other than exhaustive enumeration that can find a solution? It appears that the A* algorithm [10], a branch-and-bound technique coupled with dynamic programming and the use of heuristic underestimates, is applicable here. Again, all that is necessary is to map this problem into that formalism.

T'ne branch-and-bound part is as follows. Our goal is a covering subset of sensors with least cardinality, so we will minimize, as our A* cost, the subset size. The search naturally begins with singleton subsets, and branches at each step by extending each candidate subset with all possible individual sensors not yet included in it. Bounding--the elimination of infeasible solutions--naturally occurs when a subset grows by a useless sensor, that is, by one that does not eliminate any additional objects (i.e. one with no "new" zeros).

The dynamic programming part derives from the observation that order is unimportant in a subset, so candidate subsets (such as {2,3,1}) that have already been considered via a different developmental history (such as {1,2,3}) are eliminated.

There does not seem to be any underestimating heuristic function, however. This is because any subset can potentially become a minimum cover in just one more A* step by the addition of just the right sensor, and there is no information in the subset that can predict or deny that possibility. The true heuristic underestimate is thus uniformly zero, and the search is basically breadth first (all singleton subsets, then all doublet subsets, etc.), augmented a bit by the efficiencies of bounding and of dynamic programming. The principal advantage over blind enumeration is that subsets with useless sensors are never extended.

7.3 THE COMPLEXITY OF SEQUENTIAL SENSING

It is not hard to show that the case of sequential sensing is also NP-complete. It is clear that it is in NP since it is straightforward to verify any solution. We now show it is NP-complete because it contains parallel sensing, shown to be NP-complete, as a special case.

Let us first rephrase the sequential sensing problem in the following way. Given a cost, can we select a subset of sensors of size U, such that some permutation of the sensors achieves the goal of eliminating all interior objects (according to the semantics of the sequential match routine), at or below the cost? Now, since we can set the cost arbitrarily, let us set it to be S|j-i|, which is clearly the maximum cost possible. Since the cost is allowed to be so high, we need not worry at all about permutations; if the subset solves the problem at all, any of its permutations will also meet the cost. Thus, the problem reduces to finding a subset of size U that eliminates all interior objects; this is the parallel sensing case revisited.

Among other things, this again indicates that greedy algorithms or sorting algorithms will fail, sometimes dramatically. The following is again a minimal counterexample.

```
     i                     j
s1   X 0 0 0 0 0 0 1 1 1 1 1
s2   X 0 1 0 1 0 1 0 1 0 1 1
s3   X 1 0 1 0 1 0 1 0 1 0 1
```

A greedy algorithm--here, the equivalent of a sorting algorithm--would suggest the unit direction +<<1,1>, <2,1>, <3,1>>, that is, the sensors are to be tested in order. According to the "match" routine for sequential sensors, the sensor cost for each of the objects in order is respectively 1,1,1,1,1,1,2,3,2,3,3, for a total of 19, 8 more than the theoretic minimum of |j-i| = 11. However, the unit direction +<<2,1>, <3,1>> has sensor costs of 1,2,1,2,1,2,1,2,1,2,2, for a total of 17. As can be verified by exhaustive enumeration of the 15 possible non-empty subsequences, this is a global optimum.

Further investigation shows more of the difficulty of the problem: total sensor cost is not necessarily monotonic with sensor count. That is, the most efficient sensor subsequence need not be the one that requires the least number of sensors. Consider the following minimal example:

```
     i                     j
s1   X 0 0 0 0 0 0 0 0 1 1 1
s2   X 0 0 0 0 1 1 1 1 0 1 1
s3   X 1 ı 1 1 0 0 0 0 1 0 1
```

By inspection, the unit directive +<<2,1>, <3,1>> has sensor cost 1+1+1+1+2+2+2+2+1+2+2 = 17. However, the unit directive +<<1,1>, <2,1>, <3,1>> has sensor cost 1+1+1+1+1+1+1+1+2+3+3 = 16. Given the complexity of the problem, such anomalies should not be surprising.

The application of the A* algorithm to this problem is somewhat more complex than in the parallel sensor case, in part because an underestimating heuristic exists. Again, we identify the branch-and-bound, dynamic programming, and heuristic underestimate components.

The branch-and-bound part is as follows. Our goal is a covering subsequence of sensors with least sensor cost. The A* cost we will minimize then is solely sensor cost, since it is unrelated to subsequence length, as we have shown above. The search naturally begins with singleton subsequences, and branches at each step by extending each candidate subsequence with all possible individual sensors not yet included in it. Bounding again naturally occurs when a subsequence grows by a useless sensor, that is, by one that does not eliminate any additional objects.

The dynamic programming part derives from the curious observation that to some extent the subsequences behave as subsets, and so can be combined as the subsets were in the parallel sensor A* algorithm. Although all the permutations of a subsequence may have sensor costs that differ because of the order of the sensing, it is only necessary to keep the cheapest one. This is because, regardless of permutation, any sensor subsequence eliminates exactly the same subset of interior objects. It is only the remaining subset of interior objects that the any newly added sensor can "see".

There is a natural underestimating heuristic function, other than zero. At any given point, a developing subsequence can determine exactly how many interior objects it has yet to eliminate. At best, the next sensor added to the sequence will eliminate them all; more likely it will only eliminate a few. Thus, uneliminated object count is a true underestimate of future sensor cost.

7.4 EXAMPLE OF OPTIMIZED TRANSITION GRAPH

Suppose we complicated our example Lineland a bit by adding a second sensor (intuitively, a shape sensor), whose domain was simply {x,y}. Suppose the abstract world now looked like <bx,gy,rx,rx,ry,bx,bx,bx,gy,rx>. (Since the domains are disjoint, we have eliminated the inner level of brackets, i.e., the model really is <<b,x>,<g,y>,...>.) Applying our optimization under the sequential sensor model, we find the following transition graph, where notation has also been simplified in the obvious way: e.g. "+yr" is the equivalent of "+<<2,y>,<1,r>>", and the order of each entry is significant.

	bx	gy	rx	rx	ry	bx	bx	bx	gy	rx
bx		+g	+r		+yr	+b				
gy	-b		+r		+y	+b			+g	
rx	-b	-g		+r	+y	+b			+g	
rx	-b	-g	-r		+r	+b			+g	+r
ry	-b	-g		-r		+b			+g	+r
bx	-b	-g		-x	-r		+b		+g	+r
bx		-g		-rx	-r	-b		+b	+g	+r
bx		-g		-rx	-r		-b		+g	+r
gy		-g		-rx	-r			-b		+r
rx				-rx	-r			-b	-g	

The optimization under the parallel sensor model turns out to be identical, if one adopts the syntactic convention that "+yr" means "+{<2,y>,<1,r>}", that is, order is no longer significant.

8 FOUR MEASURES OF MAP QUALITY

We are now prepared to define map quality according to two independent criteria, and give algorithms that produce optimal custom maps according to these criteria. Since we have long since dismissed time as a valid measure of goodness, both in terms of the time cost of sensing and the time cost of motoring, and since no operation ever has any error, there are only two other measures left to minimize. They are the length of the map, and the total sensor cost. A custom map is "good" if, among all such maps that attain the goal from the starting position, it does so with the least amount of directions or the least amount of sensing. The criteria can of course be mixed. Of all the shortest maps, we can further select the blindest, and of all the blindest maps, we can speak of the shortest; usually they will not be the same map. Taken together these criteria roughly capture what a human would call the cognitive complexity of a map.

We wish, then, to investigate algorithms for eight cases of optimal map-mapping, four each under the two sensor models. The four criteria to minimize are: direction length, sensor cost, sensor cost primary and direction length secondary, and the reverse. In most of these cases, the solution is given by Dijkstra's shortest path algorithm, although what critically differentiates them is how the cost graph is defined.

8.1 PARALLEL SENSORS, MINIMIZING DIRECTION LENGTH

First we specify what is meant by direction length. Either of two definitions apply. In the simplest case, it is merely the count of the number of unit directions in the custom map; whether the subset in each POD is short or long does not matter. A more sophisticated version holds that what matters is the total length of the custom map, and so each unit direction adds a cost proportional to the size of its subset. In either case, wherever there is a unit direction in a slot of the transition graph, the cost graph is assigned the appropriate value (either 1 or U, the subset cardinality).

The solution is now straightforward: Apply Dijkstra's shortest-path algorithm to the cost graph, and read off the unit directions from the transition graph. This yields one provably optimal custom map; by the appropriate backtracking, all the other maps that are equally good can be recovered.

For example, if Lineland is modeled as the sequence discussed above, <b,g,r,r,r,b,b,b,g,r>, and the parallel-sensored navigator desires a custom map with shortest directions between 2 and 8 then the algorithm yields +g-b.

8.2 PARALLEL SENSORS, MINIMIZING SENSOR COST

In this case, wherever there is a unit direction in the transition graph, the cost graph is assigned N = U|j-i|, that is, the sensor cost of the transition under the parallel sensor model, which is the product of the cardinality of the sensor subset, U, with the distance between i and j. Dijkstra applies as before. 'ote that under this definition of map quality there is never any occasion to generate a map that overshoots the goal, as the goal would then be sensed at least twice.

In the prior example, one optimal custom map is +r+r+r+b+b+b. In fact, even with multiple sensors, one optimal custom map will always consist of |j-i| unit directions, with each unit direction consisting of a single sensor and value. This corresponds to the trivial custom map that enumerates each of the objects along the path; in effect, it measures the distance in unary notation. Note, however, that +b+b+b also is an optimal custom map here, and is shorter in terms of the number of directions.

This leads us to consider the following:

8.3 PARALLEL SENSORS, MINIMIZING DIRECTION LENGTH WITHIN SENSOR COST

The cost graph in this case is redefined to carry an ordered pair consisting of the sensory effort in the first slot and the direction length in the second. That is, cost[i,j] = (N,1) or (N,U). During Dijkstra both costs are accumulated in parallel. However, any cos. comparisons are now done lexicographically, that is, sensory effort is compared first, and any ties are broken by a comparison of direction length. The use of an ordered pair is cleaner than combining the costs by some weighting function (which of course can be used instead).

For example, under this definition of map quality, in the Lineland model of <b,g,r,r,r,b,b,b,g,r> the best custom map from 2 to 8 is given by +b+b+b, with a total cost of (6,3). In this case, it happens to be a unique solution, although it need not be in general.

Finally, for completeness we also consider:

8.4 PARALLEL SENSORS, MINIMIZING SENSOR COST WITHIN DIRECTION LENGTH

We simply reverse the cost definition to be (1,N) or (U,N); thus, in general, the order of the vector reflects the order of the nesting of the objectives. The solution, as expected, is not necessarily the same as in any of the other cases. For example, for <b,g,r,r,r,b,b,b,g,r> the best custom map from 2 to 8 is given by +g-b at a cost of (2,8), which happens to be the unique solution, with less cost than its nearest competitor, +r-b at a cost of (2,10).

8.5 SEQUENTIAL SENSORS

For all four measures of map quality, the algorithms are identical to their parallel counterparts. The values in the cost graph have the same structure as in the parallel case, but have considerably different values under the sequential sensing model. Thus, the results will also vary. (Since the above examples used only one sensor, there would be no apparent difference in our examples.) However, there is one subtle point to make.

If the measure of map quality is simply direction length, then again one has to decide the cost of a unit direction: it is either 1 or the length of the subsequence in the POD. But since it is possible under the sequential sensor model to have an optimal sensor subsequence that is actually longer than a more costly shorter subsequences (a situation impossible under parallel sensing), it is not clear whether the second definition of direction length is consistent for sequential sensing. This confusion also persists in the case where sensor cost is used to break ties among equal direction lengths. In both cases, the transitions have already been optimized for sensor cost and may be unfairly penalized by the more sophisticated definition.

9 ACKNOWLEDGMENTS

Discussions with David Laks helped formulate fundamental notions regarding landmarks in Lineland. Lane Hemachandra sketched the first NP-completeness proofs, which we were able to streamline. David Eppstein shared many insights about the complexity of other problems similar to navigation. Monnett Hanvey provided us with a solid grounding in the existing literature of navigation by actual robots.

References

1. Edwin Abbott. *Flatland: A Romance of Many Dimensions.* Dover Publications, 1952.

2. Aho, Upcroft, and Ullman. *Design and Analysis of Computer Algorithms.* Addison-Wesley, 1974".

3. Ernest Davis. *Representing and Acquiring Geographic Knowledge.* Morgan-Kaufmann, 1986.

4. Thomas Dean. A Taxonomy of Map Learning Problems. CS-87-16, Department of Computer Science, Brown University, August, 1987.

5. Michael R. Garey, David S. Johnson. *Computers and Intractability.* W. H. Freeman, 1979.

6. H. A. Gleason Jr.. *Linguistics and English Grammar.* Holt, Winehart, and Winston, NY, 1979.

7. Richard C. Karp. Reducability Among Combinatorial Problems. In *Complexity of Computer Computation*, Plenum Press, New York, NY, 1972, pp. 85-103.

8. Benjamin Kuipers. "Modeling Spatial Knowledge". *Cognitive Science 12* (1978), 129-153.

9. Monnett Hanvey. Enviromental Representation for Mobile Robot Navigation. Department of Computer Science, Columbia University, January, 1988.

10. Nils D. Nilsson. *Principles of AI.* Tioga Press, Palo Alto, California, 1980.

11. Lynn A. Streeter, Daine Vitello, Susan W. Wonsiewicz. "How to tell people where to go: comparing navigational aids". *Int. J. Man-Machine Studies 22* (1985).

3-D VISION FOR OUTDOOR NAVIGATION BY AN AUTONOMOUS VEHICLE

Martial Hebert **Takeo Kanade**

Carnegie Mellon University The Robotics Institute
5000 Forbes Avenue, Pittsburgh PA 15213

Abstract

This paper reports progress in range image analysis for autonomous navigation in outdoor environments. The goal of our work is to use range data from an ERIM laser range finder to build a three-dimensional description of the environment. We describe techniques for building both low-level description, such as obstacle maps or terrain maps, as well as higher level description using model-based object recognition. We have integrated these techniques in the NAVLAB system[1].

1. Introduction

Research in robotics has recently focused on the field of autonomous vehicles, that is mobile units that can navigate under computer control based upon sensory information. Several components are involved in the design of such a system. A high level cognitive module is in charge of making decision based on the perceived environment and the mission to be carried out. Sensory modules, such as video image analysis or range data analysis, transform the sensors' output into a compact description that can be used by the decision-making modules. Low-level control software converts decisions into actions performed by the hardware.

In this paper, we focus on one type of sensory component, the analysis of range data for an autonomous vehicle navigating in an environment with features such as trees, uneven terrain, and man-made objects. In particular, we study the use of the Environmental Research Institute of Michigan (ERIM) laser range finder to perform four tasks: obstacle detection, surface description, terrain map building, and object recognition (Fig. 1-1). Obstacle detection is the minimum capability required by an autonomous vehicle in order to navigate safely. Surface description is needed when the obstacle detection is not sufficient for safe navigation, in the case of uneven terrain for example, or when a more accurate description of the environment is needed, *e.g.* for object recognition. Terrain map building is the process by which surface descriptions from different vantage points are accumulated in a consistent map. Object recognition capabilities are required when the vehicle must recognize and locate known landmarks, *e.g.* a traffic sign, in order to carry out its mission.

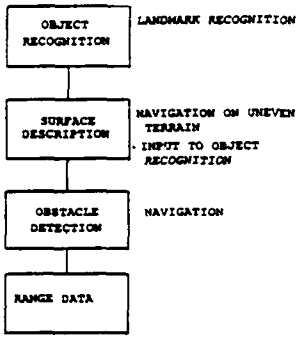


Figure 1-1: Range data processing

2. Intermediate representations of range data

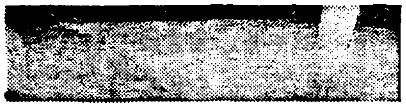
2.1. Sensor data

The ERIM sensor derives the range at each point by measuring the difference of phase between a modulated laser beam and reflection from the scene. A two-mirrors scanning mechanism directs the beam onto the scene so that an image of the scene is produced. In the ERIM ALV version, the field of view is ±40 in the horizontal plane and 30 in the vertical plane, from 15 to 45 . The resulting range image is a 64 × 256 8-bit image. The frame rate is currently two images per second. The nominal range noise is 0.4 feet at 50 feet. The sensor also produces reflectance images in which the value of each pixel is the amount of light reflected by the target. Figure 2-1 shows an example of a range image and the corresponding reflectance image.

This research was sponsored in part by DARPA monitored by the U.S. Army Engineer Topographic Lab under contract DACA76-85-C-0003, by DARPA monitored by the Air Force Avionics Laboratory under contract F33615-87-C-1499, and by NSF under contract DCR 8604199.

The ERIM sensor presents some limitations: Since only the phase difference is measured, the range values are known only *modulo* 64 feet. This causes ambiguity in the range data. The resolution degrades rapidly as the range increases. This is due to the divergence of the beam, which produces larger footprints as the distance increases, and to the scanning mechanism. Since the scanning angles are discretized using constant increments, the density of points decreases as the range increases. Points may be incorrectly measured at the edges of objects due to multiple reflections of the beam. This effect is common to all active scanning techniques and is known as the "mixed points" problem. We have found that applying a median filter to the original image eliminates most mixed points.

(a) Range image: the darker pixels are closest.



(b) Reflectance image.

Figure 2-1: An example of range and reflectance images

2.2. Vehicle centered coordinates

The raw data from the ERIM scanner represent range as a function of angles θ and ϕ of the two mirrors.

As shown in Figure 2-2, we assume that a coordinate frame $\vec{u}, \vec{v}, \vec{w}$ is attached to the scanner. We use another coordinate frame, the "vehicle" frame, $\vec{i}', \vec{j}', \vec{k}'$ to express the measured points so that the resulting values are vehicle-centered and are therefore independent of the sensor configuration. We can thus derive the coordinates (x, y, z) of the point measured at pixel (row, col) with range D. If ϕ is the angle between $(\vec{u}, \vec{v})$ and the direction of the measured beam $\vec{d}$ at pixel i,j, θ is the angle between $(\vec{u}, \vec{w})$ and the direction of the measured beam $\vec{d}$ at pixel i,j, ϕ_s is the starting vertical scanning angle, $\Delta\theta$ and $\Delta\phi$ are the angular increments, and τ is the tilt angle of the scanner, that is the angle between the planes $(\vec{i}', \vec{j}')$ and $(\vec{u}, \vec{v})$ as shown in Figure 2-2, then the conversion is:

$$\theta = (j - 128) \times \Delta\theta$$
$$\phi = \phi_s - i \times \Delta\theta$$

$$x = D(\cos\theta(\cos\tau\cos\phi - \sin\tau\sin\phi))$$
$$y = D\sin\theta$$
$$z = D(\cos\theta(\sin\tau\cos\phi + \cos\tau\sin\phi))$$

Figure 2-3 shows the data of the image of Figure 2-1 after conversion to vehicle coordinates as viewed in the direction of the $\vec{w}$.

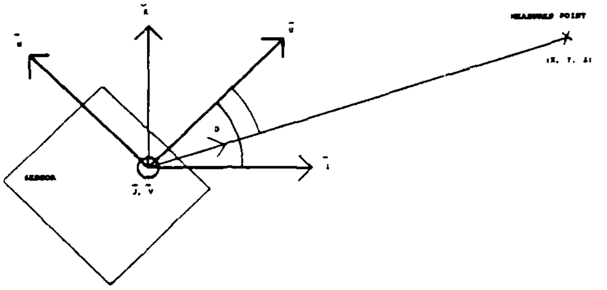


Figure 2-2: Conversion to vehicle coordinates

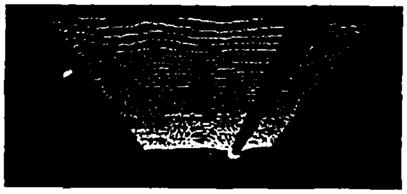


Figure 2-3: Overhead view of the data of Figure 2-1

2.3. Bucket map

In outdoor environments, the ground plane ($\vec{i}, \vec{j}$) has a privileged role: the terrain can be modeled as a function $z=f(x,y)$, x and y being the coordinates on the ground plane. In order to take advantage of the ground plane, we used a intermediate representation, the bucket map.

A bucket map is defined by a regular grid on a reference horizontal plane. Each cell of the grid, or bucket, contains a set of measured points as shown on Figure 2-4. The points within a bucket may all be from the same image or from several consecutive images. The size of a bucket is typically 30 cms x 30 cms.

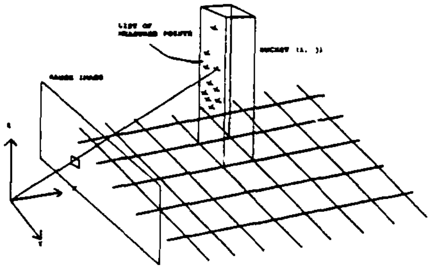


Figure 2-4: The bucket map structure

3. Obstacle detection

The first task of range data analysis for navigation is to report the portions of the environment that are potentially hazardous. We must identify individual objects in the environment that the vehicle must avoid. Most obstacle detection algorithms combine surface discontinuities and surface slope[2, 3] to extract untraversable regions in the image using a vehicle model[4]. Faster algorithms use apriori knowledge of the terrain, *e.g.* flat ground assumption, by computing the difference between the range image and the expected ideal image[5]. Since we want to be able to navigate in a variety of environments, we chose the first approach which, although computationally expensive, allows us to handle uneven terrains. Specifically, we identify points in the bucket map at which the elevation exhibits a large discontinuity and points at which the surface slope is above a given threshold. The first set of points corresponds to the edges of the objects, the second lies within the surface of an object facing the sensor. The obstacle detection algorithm is divided into four steps:

1. Detect elevation discontinuities on the bucket map. The discontinuities are computed in 2x2 windows around each point;
2. Compute the surface normal for each bucket;
3. Detect the buckets for which the surface normal whose angle with the vertical direction is greater than a given threshold;
4. Extract connected regions from the set of buckets detected at step 3 so that each region corresponds to an object. Two buckets are connected if they do not cross the line of elevation discontinuity.

The rationale for using two criteria, elevation and surface normals, is that the surface normals are meaningless at the edge of an object, and the elevation cannot be used alone without some knowledge of a ground-plane on which objects are known to rest. One possible undesirable result of the detection algorithm is that a small portion of the terrain might be reported as an obstacle due to the resolution of the bucket map. This error can be corrected only when a more complete object description is created. It does not, however, significantly affect the behavior of the vehicle since only small regions are involved.

For the purpose of obstacle avoidance, the detected objects are represented by polygons on the ground surface in order to be used by a path planner. Figure 3-1 shows a range image, the location of the buckets classified as parts of objects, and the polygonal representation of the obstacle map. The squares indicate the buckets in which the objects have been found. The large polygon enclosing the map is the boundary of the portion of the environment seen by the sensor.

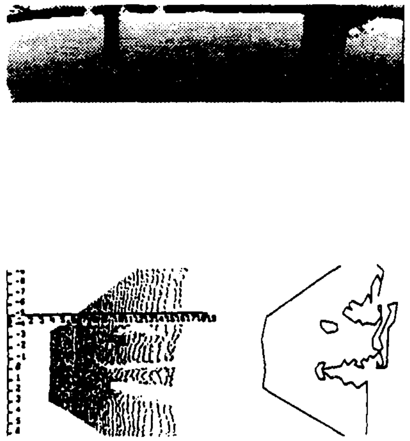


Figure 3-1: Obstacle Detection

4. Surface description

The obstacle detection algorithm is sufficient for vehicle navigation in a simple environment which includes only a smooth terrain and discrete obstacles. A typical example of such an environment occurs in road following applications. We need a more sophisticated representation in two cases:

- The surrounding terrain is uneven. In that case, part of the environment that may be hazardous or costly to navigate cannot be described as discrete objects.

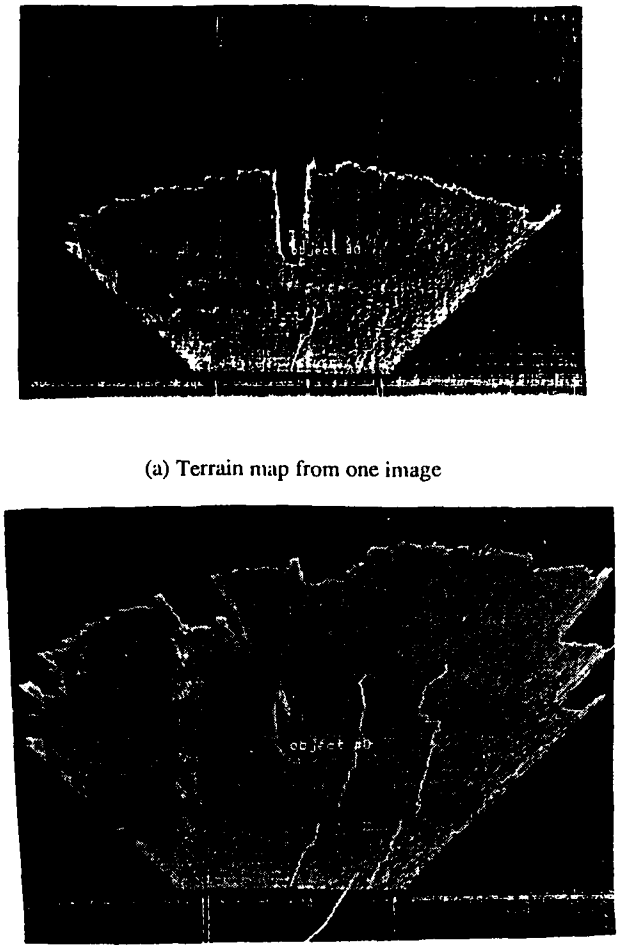
The strategy for expanding a region is to merge the best point at the boundary at each step. This strategy guarantees a near optimal segmentation. It has, however, two major drawbacks however: it may be computationally expensive, and it may lead to errors due to sensor errors on isolated points, such as mixed points. To alleviate these problems, we use a multi-resolution approach. We first apply the segmentation to a reduced image in which each pixel corresponds to a $n \times n$ window in the original image, n being the reduction factor. This first, low-resolution, step produces a conservative description of the image (Fig. 4-1.c). The low-resolution regions are then expanded using the full-resolution image (Fig. 4-1.d). No new regions are created at full resolution. Figure 4-2 shows the segmentation of an image of uneven terrain.

In addition to the pure region segmentation, we use the edges extracted from the reflectance image to improve the description. In the low-resolution segmentation step, pixels that correspond to a window that contains at least one edge pixels are removed. In the full-resolution step, regions are expanded so that they do not cross an edge. As a result, edge pixels are all part of the regions boundaries. Explicitly including edges improves the segmentation in two ways: First, edges that correspond to low-amplitude occluding edges separates regions that may be merged in the range image segmentation. Second, reflectance edges can delineate surface markings that are not visible in the range image. Figure 4-1.b shows an edge image obtained by applying a 10×10 Canny edge detector.

5. Terrain map building

Map building is the process of combining observations of an unknown terrain into a coherent representation. Building a terrain map serves two purposes: It allows to report a set of observations as a product of an exploration mission, and it improves the performance of an autonomous vehicle when the vehicle traverses a previously mapped region. Two types of information are kept in a terrain map: the low-level measurements that are accumulated in a bucket map as described in Section 2.3, and the terrain or object features.

The main issue in the map building process is the matching and enhancement of a map built from measurements acquired from different vantage points. This issue presents some challenging aspects. Since we do not put any constraints on the vehicle's trajectories, observations of the environment may be radically different from one observation to the other. This requires the identification of common features between observations, as well as the use of uncertainty on the various vehicle position estimates. The insertion of a new frame in the map must therefore proceed in three steps: First, the current estimate of the vehicle position is used to predict matchings between the current map and the new observed features. Second, a new position estimate is computed based on the matchings and the current position estimate. Third, the map is updated by inserting the new observations. This involves the insertion of new measurements in the grid representation, the insertion of the new features in the map, and the updating of existing map features that have been matched with newly extracted features.



(a) Terrain map from one image

(b) Terrain map from four images

Figure 5-1: Terrain map building (including road features)

We have included the map building techniques in the Carnegie-Mellon NAVLAB system. A terrain map was maintained over a hundred meters while the vehicle was running autonomously under control of the road following program. The current vehicle estimate was used as an initial estimate for the matching. Due to memory limitations, only the portion of the map within a window centered at the current vehicle was kept in the system. The features used for the matching part in that implementation are: the primitives of the surface description (as described in Section 3), the location of discrete objects (as described in Section 2), and location of the road edges extracted from the reflectance channel. The features are weighted according to their uncertainty, for example the variance σ_a in the case of planar features. The road edges are a special type of features: since road detection from reflectance is currently not reliable, the edges are used only if the contrast in the current reflectance image, *e.g.* the strength of the detected road edges, is high enough.

6. Object recognition

6.1. Recognition strategy

The goal of an object recognition algorithm is to find the most consistent interpretation of a scene given a stored list of primitives $(M_{i1},..,M_{in})$, the model, and a segmentation of the observed scene, $(S_{j1},..,S_{jn})$. The algorithm must therefore search all the possible matchings $((M_{i1},S_{j1}),..,(M_{in},S_{jn}))$. This search being an combinatorial problem, the main issue is to prune the search space in order to be able to process complex scenes. Many strategies have been proposed for solving the object recognition problem[6]. The most common approach is to use geometric constraints to constrain the search, assuming the objects are rigid. This approach may require an accurate geometric model which is usually not available in our application. Another approach is to generate beforehand the possible appearances of the object to recognized in order to reduce the search space by precompiling the constraints in the model[7, 8]. We use a combination of both approaches in which geometric constraints are precompiled in the model. The model has two components: a set of surface patches, M, and a set of constraints, C. The constraints encapsulate knowledge about the object's shape, such as "surfaces M_i and M_j are orthogonal". A constraint, c, associated with a set of regions $(M_{i1},..,M_{in})$ can be viewed as a function that decides whether a partial matching $((M_{i1},S_{j1}),..,(M_{in},S_{jn}))$ is acceptable. The number n of regions involved may be different depending on the constraint. For example the constraint on the area of a region is a unary constraint, while the orthogonality constraint is a binary constraint. The list of constraints and their implementation are discussed in the next two Sections.

The search algorithm first constructs a list of candidates for each model region, M_i, by applying the unary constraints associated with M_i to every scene region, S_j. This provides a first reduction of the search space according to unary constraints. The algorithm then explores the remaining search space, discarding the partial solutions that do not satisfy the remaining constraints. In other words, each time a new pairing, (M_i,S_j), is added to a partial solution, $((M_{i1},S_{j1}),..,(M_{in},S_{jn}))$, the constraints associated with M_i are evaluated over the new set of pairings. The partial solution is not explored further if one of the constraints is not satisfied. The result of all the constraint evaluations are stored in tables, so that a constraint is never evaluated twice on the same set of pairings.

The result of the search algorithm is a small set of solutions. The last step of the recognition algorithm is to compute a score that reflects the quality of each solution. This last step is necessary since there is no way of forcing the search to produce only one solution because of near-symmetries in the model, segmentation errors, or even the presence of several instances of the object in the scene. The actual computation of the score is discussed in detail in Section 5.3. The next sections describe the details of the algorithm. We use a car as an example of object model in discussing the algorithm.

6.2. Constraints

The constraints we currently use are:

- ***NX, NY, NZ***: constrains the components of the surface normal of a region. This constraint is used to implement natural limitations on the orientation of an object, such as "the roof of a car cannot be vertical".
- ***Z***: constrains the vertical position of a region.
- ***AREA***: constrain the area of one region.
- ***ANGLE***: constrains the angle between two regions.
- ***NEIGHBOR***: constrains two regions to be neighbors by computing the distance between the boundaries of two regions.
- ***EXCLUDE***: Forbids two regions to be visible at the same time. This constraint is based on the notion of aspects[7, 9]. An aspect is a set of regions that can be observed from a given viewpoint. Instead of explicitly enumerating the possible aspects of an object, the ***EXCLUDE*** constraint describes them implicitly.
- ***DISTANCE***: constrains the distance between two surfaces.

The constraints are precomputed and stored in the model. Each constraint is described by the following structure:

- **number of arguments,** N: For example, the constraint ***ANGLE*** which constrains the angle between two regions has two arguments. The maximum number of arguments is currently three.
- **evaluation function,** F: A function that returns an interval given N regions. For example, the constraint ***ANGLE*** computes an interval centered around the angle between two input regions. The size of the interval is determined at run-time by the uncertainty on the parameters of the regions. In the case ***ANGLE***, the interval width is given by the angular variance σ_a within the two input regions.
- **interval,** I: An interval, or set of intervals, which must intersect the computed interval to satisfy the constraint.

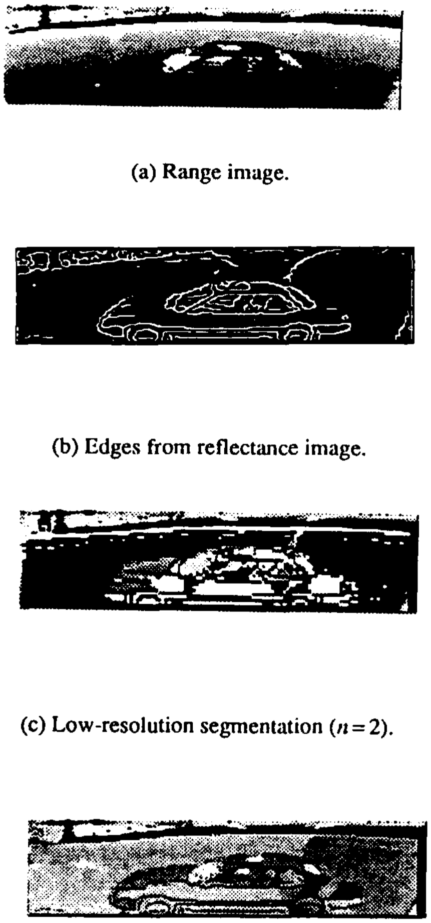
This representation of constraints is flexible: A new type of constraint can be easily added to a model by simply defining the appropriate evaluation function. Building a new model is easier since the constraints are not hardcoded in the recognition program.

6.3. Evaluation of the solutions

One would like to have a recognition program that generates only one solution that is reported as the recognized object in the scene. Unfortunately, the search algorithm generates many solutions that have to be evaluated in order to determine the best one. There are three reasons why the search generates multiple solutions: First, the constraints we

- The mission requires the recognition of specific objects given apriori models. In that case, the mere knowledge of the existence and position of an object in the world is not sufficient, we need a more detailed description of its shape.

We describe surfaces by a set of connected surface patches. Each patch corresponds to a smooth portion of the surface and is approximated by a parameterized surface. In addition to the parameters and the neighbors, each region has two uncertainty factors: σ_a, and σ_d. σ_a is the variance of the angle between the measured surface normal and the surface normal of the approximating surface at each point. σ_d is the variance of the distance between the measured points and the approximating surface. Those two attributes are used in the object recognition algorithm.



(a) Range image.

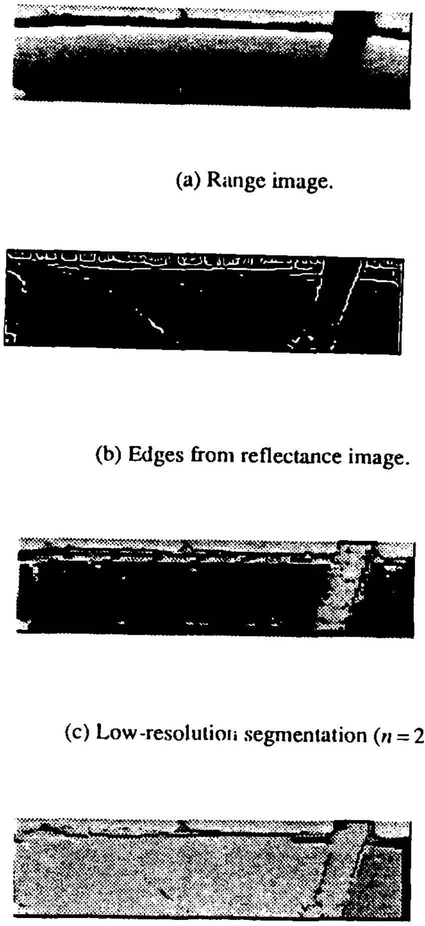
(b) Edges from reflectance image.

(c) Low-resolution segmentation ($n=2$).

(d) Final segmentation.

Figure 4-1: Range image segmentation

The surface description is obtained by segmenting the range image into regions. Several schemes for range image segmentation have been proposed in previous work[6]. These techniques are based either on clustering in some parameter space, or region growing using smoothness criteria of the surface. We chose to combine both approaches into a single segmentation algorithm. The algorithm first attempts to find groups of points that belong to the same surface, and then uses these groups as seeds for region growing, so that each group is expanded into a smooth connected surface patch. The smoothness of a patch is evaluated by fitting a surface, plane or quadric, in the least-squares sense.



(a) Range image.

(b) Edges from reflectance image.

(c) Low-resolution segmentation ($n=2$).

(d) Final segmentation.

Figure 4-2: Range image segmentation

use are very liberal so that the same model can be used for a wide range of scenes, consequently false solutions are difficult to avoid. Second, the object may have near-symmetries that lead to several equally valid interpretations. Third, the image segmentation being imperfect, an object region may be broken into several pieces in the image, thus producing several equivalent solutions.

Our approach to evaluating solutions is to first compute the position and orientation, or pose, of the object for each solution, to then generate a synthesized, or *predicted*, range image using the estimated pose, and to finally correlate the predicted image and the original range image. We derive a score from the correlation measure between the two images which is used to discard erroneous solutions, and to sort the other solutions.

The pose is calculated for each solution by minimizing the two sums:

$$\sum_{(i,j)} area_i \| \boldsymbol{R} \vec{n}_i^{\,model} - \vec{n}_j^{\,scene} \|^2$$

and

$$\sum_{(i,j)} area_i \| \boldsymbol{R} \vec{c}_i^{\,model} + \vec{t} - \vec{c}_j^{\,scene} \|^2$$

where $\boldsymbol{R}$ and $\vec{t}$ are the estimated rotation and translation, $\vec{n}_i^{\,model}$ and $c_i^{\,model}$ (resp. $\vec{n}_j^{\,scene}$ and $c_j^{\,scene}$) are the surface normal and center of region i (resp. j) of the model (resp. scene). The summation is over all the pairs (scene region j, model region i).

In order to compute the predicted image, we have to compute the position of each point of the model in image coordinates, as well as the predicted range. The position in the predicted range image of a point $\vec{p}$ on the surface of the model is given by:

$$\phi^{predicted} = \text{atan}\ (x/z)$$

$$\theta^{predicted} = \text{atan}\ (y \times \cos\ (\phi))/x)$$

$$row = (\phi_s - \phi^{predicted})/\Delta\phi$$

$$col = (\theta^{predicted} - \theta_s)/\Delta\theta$$

$$d^{predicted} = \sqrt{x \times x + y \times y + z \times z}$$

In this equation: x, y, z are the coordinates of the transformed point $\boldsymbol{R}\vec{p} + \vec{t}$, (*row*, *col*) is the predicted location is the image, and $d^{predicted}$ is the predicted range value at that location.

The correlation between predicted and actual range images is given by:

$$C = \sum_{(i,j) \in P \cap O} a_{ij} \times | d_{ij}^{predicted} - d_{ij}^{original} |$$

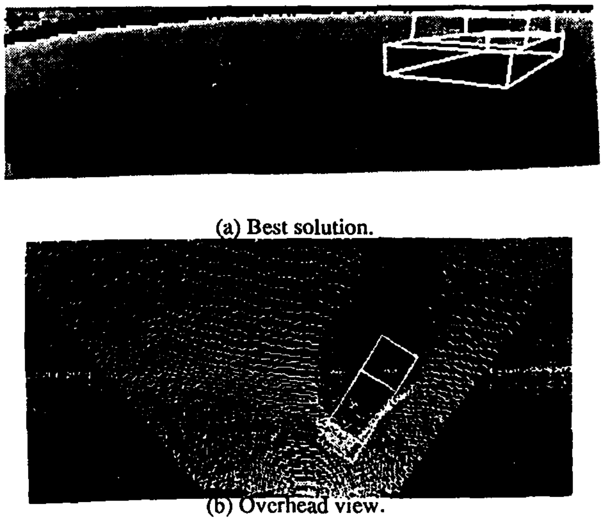
where a_{ij} is the area intercepted by pixel i,j, and the summation is made over the intersection $\boldsymbol{P} \cap \boldsymbol{O}$ of the object in the predicted image, $\boldsymbol{P}$, and the object in the observed range image, $\boldsymbol{O}$.

The correlation must be normalized to obtain a score S that is between 0 and 1:

$$S = area(\boldsymbol{P} \cap \boldsymbol{O}) / area(\boldsymbol{P} \cup \boldsymbol{O})$$

$$\times (1 - C/(\boldsymbol{K} \times area(\boldsymbol{P} \cap \boldsymbol{O})))$$

In this equation, $\boldsymbol{K}$ is the maximum range difference allowed between predicted and observed images (independent of the image). S is normalized by $area(\boldsymbol{P} \cap \boldsymbol{O}) / area(\boldsymbol{P} \cup \boldsymbol{O})$ to avoid problems when $\boldsymbol{P} \cap \boldsymbol{O}$ is very small, in which case we would give a high score to a very poor solution. We use the score S to eliminate false solution (typically $S < 0.5$), and to sort the solutions by decreasing score. Figure 6-1 shows the solution of highest score found on one image, the top image shows the superimposition of the recognized model and the range image, the bottom image is the overhead view of the superimposition.



(a) Best solution.

(b) Overhead view.

Figure 6-1: Object recognition

6.4. Results

We have tested the object recognition program on 23 images of two different objects, each image corresponds to a different aspect of the objects, or to a different distance between the object and the sensor. Figure 6-2 shows a sample of the set of images. The failure modes of the program are as follows:

- In two cases, the program failed to produce any interpretation. These cases occur when not enough regions have been extracted to perform the recognition.

- In two cases, the program produced a set of interpretations, but the correct solution was not in the top-score set of solutions.

- In three cases, the program produced the correct interpretation as part of the top-score set of solutions but not as the best solution. These cases occur when a near symmetry in the image cannot be disambiguated based on the shape information only, as a result, both the correct solution and its symmetrical have the same score.

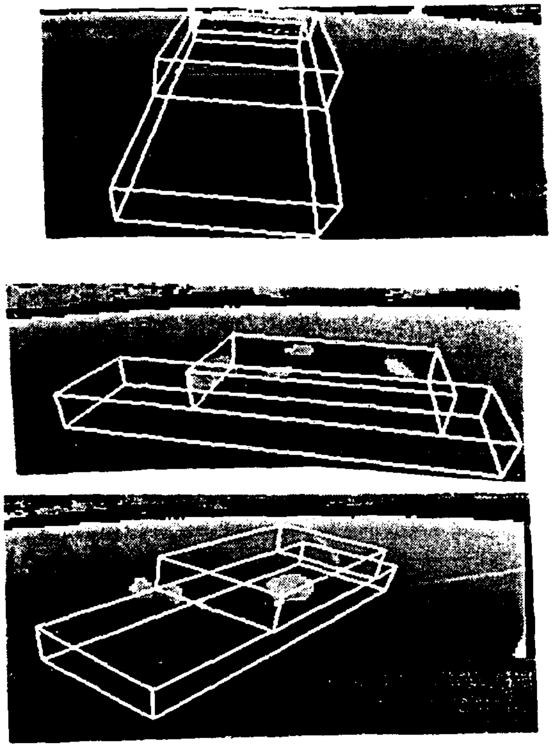


Figure 6-2: Object recognition on a sample set of images

The object recognition algorithm can be improved in several ways: The recognition of one object requires an average of 1,500 constraint evaluations, most of which are rarely used for rejecting a partial solution. One optimization is to order the constraints in the model so that the constraints most likely to prune the search are evaluated first.

The scoring procedure leads to very close scores for symmetric or ambiguous solutions. As an example, Figure 6-3 shows two symmetric solutions for which the difference between the two scores is below 1%. This can be improved by including other sources of information, such as video or reflectance images, in the scoring procedure.

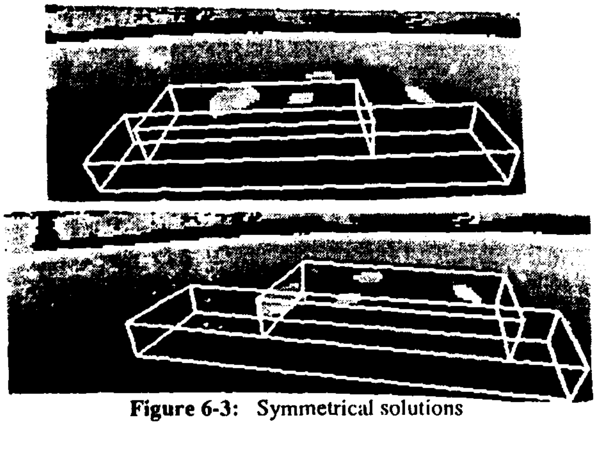


Figure 6-3: Symmetrical solutions

7. Conclusion

We have presented a set of techniques for producing 3-D environment descriptions for an autonomous vehicle. Those techniques and their output descriptions have been designed to fulfill the requirements of the major tasks of an autonomous vehicle such as obstacle avoidance, landmark recognition, and map building. We have conducted an numerous demonstrations in the CMU autonomous vehicle system, the NAVLAB, for navigation and map building applications.

We are currently pursuing further research along three lines: implementation on faster hardware, handling uncertainty, and developing a more general scheme for object recognition.

The primary bottleneck in the development of an autonomous vehicle is the computation time required by the sensory modules such as the range image analysis module. We are in the process of porting the algorithms to a fast systolic machine, the WARP[10]. The obstacle detection module has been already successfully demonstrated with a cycle on the order of one second on the WARP. The second line of work is the representation of uncertainty. Sensor measurements, vehicle position estimates, and segmentation process induce errors in the final 3-D description. These errors can be quantified and taken into account in all the range analysis algorithms. We plan to use an explicit representation of the uncertainty building surface description[11, 12]. Third, we plan to develop a more general scheme for object recognition. Third, we plan to apply the object recognition algorithm to a larger class of objects and algorithms.

References

1. C. Thorpe, M. Hebert, T. Kanade, S. Shafer, *Vision and navigation for the Carnegie-Mellon Navlab*, Annual Reviews Inc., 1987.

2. M. Hebert, T. Kanade, "First results on outdoor scene analysis", *Proc. IEEE Robotics and Automation*, San Francisco, 1985.

3. D. Y. Tseng, M. J. Daily, K. E. Olin, K. Reiser, F. M. Vilnrotter, "Knowledge-based vision techniques annual technical report", Tech. report ETL-0431, U.S. Army ETL, 1986.

4. M. J. Daily, J. G. Harris, K. Reiser, "Detecting Obstacles in Range Imagery", *Image Understanding Workshop*, Los Angeles, 1987.

5. R. T. Dunlay, D. G. Morgenthaler, "Obstacle detection and avoidance from range data", *Proc. SPIE Mobile Robots Conference*, Cambridge, MA, 1986.

6. P. J. Besl, R. C. Jain, "Three-dimensional Object Recognition", *ACM Comp. Surveys*, Vol. 17, No. 1, march 1985.

7. K. Ikeuchi, "Precompiling a geometrical model into an interpretation tree for object recognition in bin-

picking tasks'', *Image Understanding Workshop*, Los Angeles, 1987.

8. C. Goad, ''Special purpose automatic p.ogramming for 3D model-based vision'', *Proc. Image Understanding Workshop*, 1983.

9. M. Hebert, T. Kanade, ''The 3D Profile method for object recognition'', *Proc. Computer Vision and Pattern Recognition*, San Francisco, 1985.

10. Webb, H. and Kanade, T., *Vision on a Systolic Array Machine*, Academic Press, 1985.

11. L. Matthies, S. A. Shafer, ''Error Modelling in Stereo Navigation'', Tech. report CMU-RI-TR-86-140, Carnegie-Mellon University, the Robotics Institute, 1986.

12. R. M. Bolle, D. B. Cooper, ''On Optimally Combining Pieces of Information, with Application to Estimating 3-D Complex-Object Position from Range Data'', *PAMI*, Vol. 8, No. 5, september 1986.

MACHINE-INDEPENDENT IMAGE PROCESSING: PERFORMANCE OF APPLY ON DIVERSE ARCHITECTURES

Richard S Wallace[1,2], Jon A Webb[1], and I-Chen Wu[1]

[1]Department of Computer Science
Carnegie Mellon University
Pittsburgh, Pennsylvania 15217

[2]Hughes Artificial Intelligence Center
Calabasas, California 91302

ABSTRACT

One of the most important obstacles standing in the way of widespread use of parallel computers for low-level vision is the lack of a programming language that can be mapped efficiently onto different computer architectures which is suited for low-level vision. We present such a language, and demonstrate its capabilities by comparing the performance of the Hughes HBA, the Carnegie Mellon Warp machine, and the Sun 3 on a large set of low-level vision programs. In order to demonstrate its efficiency, we also compare performance of the Sun code with routines of similar function written by professional programmers.

1. Introduction

Low-level vision is an area of computer science that is ripe for the use of parallel computers. This class of operations is easily parallelizable. Indeed, many parallel computers are already being developed for use at this level of vision. These computers offer enormous speedup to the developer of computer vision algorithms, since these operations are so time-consuming, but software development is necessary before they can be used.

In pervious work, Hamey, Webb, and Wu developed a language called Apply [2] which can generate efficient programs for a variety of parallel machines given a single source code. Apply therefore allows machine independent programming, for a limited, application-specific, set of algorithms.

Apply has been used to develop a library of vision programs called WEB, which includes routines for many low-level vision operations. Over 130 programs exist in WEB, 80% of which are written in Apply. The Apply routines include basic image operations, convolution, edge detection, smoothing, binary image processing, color conversion, pattern generation, and multi-level image processing. This library is therefore a machine-independent software base for low-level image processing.

Because of the machine independence of the Apply language, programs written in Apply can be ported from one machine to another simply by recompilation. Moreover, the Apply compiler and the WEB library allow the comparison of the performance of vision machines, since the same source code will be running on both machine, which is the strongest possible basis for comparison of two computers.

In this paper, we demonstrate this by studying the performance of Apply on three diverse architectures, by examining the execution times of programs from WEB. The architectures are the Carnegie Mellon Warp machine, a 100 MFLOPS systolic array machine [1]; a Sun 3/75 workstation; and the Hughes Aircraft Corporation Hierarchical Bus Architecture (HBA) [6], a MIMD computer specifically designed for image processing applications. These architectures differ in the number of processors, in the processor topology, and in the underlying processor, but Apply generates efficient code for all of them.

We establish a baseline of Apply's performance by comparing Apply code with code generated by hand for some of the computers. Then we use

processing co-ordinates.

Constant parameters may be scalars, vectors or two-dimensional arrays. They represent precomputed constants which are made available for use by the procedure. For example, a convolution program would use a constant array for the convolution mask.

execution time as a basis for evaluating the performance of Apply, and for studying the suitability of these machines as image processors.

2. The Apply Language

Apply implements a simple *data partitioning* model for image processing that makes explicit the parallelism of low-level vision algorithms. When using Apply, the programmer writes a procedure which defines the operation to be applied at a particular pixel location. The procedure conforms to the following programming model:

- It accepts a window or a pixel from each input image.
- It performs an arbitrary computation without side-effects.
- It returns a pixel value for each output image.

The idea of the Apply programming model grew out of a desire for efficiency combined with ease of programming for a useful class of low-level vision operations. After implementing Apply, the following additional advantages became evident:

- Apply concentrates programming effort on the actual computation to be performed instead of the looping in which it is embedded. This encourages programmers to use more efficient implementations of their algorithms. For example, a Sobel program gained a factor of four in speed when it was reimplemented with Apply. This speedup primarily resulted from explicitly coding the convolutions. The resulting code is *more comprehensible than the earlier implementation.*
- Apply programs are easier to write, easier to debug, more comprehensible and more likely to work correctly the first time. A major benefit of Apply is that it greatly reduces programming time and effort for a very useful class of vision algorithms. The resulting programs are also faster than the programmer would probably otherwise achieve.

2.1. Details of the language

Apply is designed for programming image to image computations where the pixels of the output images can be computed from corresponding rectangular windows of the input images. The essential feature of the language is that each operation is written as a procedure for a single pixel position. The Apply compiler generates a program which executes the procedure over an entire image. No ordering constraints are provided for in the language, allowing the compiler complete freedom in dividing the computation among processors.

Each procedure has a parameter list containing parameters of any of the following types: *in, out* or *constant.* Input parameters are either scalar variables or two-dimensional arrays. A scalar input variable represents the pixel value of an input image at the current processing co-ordinates. A two-dimensional array input variable represents a window of an input image. Element (0,0) of the array corresponds to the current processing co-ordinates.

Each output variable represents the pixel value of an output image. The final value of an outout variable is stored in the output image at the current

2.2. An Implementation of Sobel Edge Detection

As a simple example of the use of Apply, let us consider the implementation of Sobel edge detection. Sobel edge detection is performed by convolving the input image with two 3×3 masks. The horizontal mask measures the gradient of horizontal edges, and the vertical mask measures the gradient of vertical edges. Diagonal edges produce some response from each mask, allowing the edge orientation and strength to be measured for all edges. Both masks are shown in Figure 1.

```
|  1  2  1 |   | 1  0 -1 |
|  0  0  0 |   | 2  0 -2 |
| -1 -2 -1 |   | 1  0 -1 |
```

Horizontal *Vertical*

Figure 1: The Sobel Convolution Masks.

An Apply implementation of Sobel edge detection is shown in Figure 2.

```
procedure sobel
   (inimg  : in array (-1..1, -1..1) of byte
               border 0,
    thresh : const real,
    mag    : out real)
is
      horiz, vert : integer;
begin
      horiz := inimg(-1,-1) + 2 * inimg(-1,0)
            + inimg(-1,1) - inimg(1,-1)
            - 2 * inimg(1,0) - inimg(1,1);
      vert := inimg(-1,-1) + 2 * inimg(0,-1)
            + inimg(1,-1) - inimg(-1,1)
            - 2 * inimg(0,1) - inimg(1,1);
      mag := sqrt(REAL(horiz)*REAL(horiz)
                + REAL(vert)*REAL(vert));
      if mag < thresh then
          mag := 0.0;
      end if;
end sobel;
```

Figure 2: An Apply Implementation of Thresholded Sobel Edge Detection.

First the input, output and constant parameters to the function are defined. The input parameter **inimg** is a window of the input image. The constant parameter **thresh** is a threshold. Edges which are weaker than this threshold are suppressed in the output magnitude image, **mag**. **Horiz** and **vert** which are internal variables used to hold the results of the horizontal and vertical Sobel edge operator.

The input image window is also defined in the declaration of inimg. It is a 3×3 window centered about the current pixel processing position, which is filled with the value 0 when the window lies outside the image. This same line declares the constant and output parameters to be floating-point scalar variables.

The computation of the Sobel convolutions is implemented by the straightforward expressions in the body of the function. These expressions are readily seen to be a direct implementation of the convolutions in Figure 1.

3. The WEB Library

Apply has been used to implement a large portion of the WEB library of vision programs, which is a large library of vision programs implemented for use on the Carnegie Mellon Warp machine. The original purpose of the library was to facilitate vision programming on the Warp machine.

WEB currently consists of over 130 routines, 80% of which are written in Apply. The rest are written in W2, which is the standard Warp programming language. All of the local image-to-image vision routines in WEB are written in Apply; the W2 routines include non-local routines such as histogram, image warping, and connected components.

WEB is based on the SPIDER library of FORTRAN programs [5]. This is a subroutine library, developed in Japan, for image processing using FORTRAN. Routines from SPIDER will be compared here in performance with equivalent routines from WEB in order to measure Apply's performance as a code generator for Sun.

4. Implementations of Apply

Apply has been implemented, so far, on three architectures: the Carnegie Mellon Warp machine, the Sun 3/75 workstation, and the Hughes HBA. In all three cases, Apply is being used regularly for programming low-level vision operations and has been a useful tool in making these operations easier to write. We describe the implementations in order to give examples of how Apply can be implemented, and to show how the Apply implementor can make use of special features of the architectures to make Apply more efficient.

4.1. Apply on Warp

We implement Apply on Warp by the *input partitioning* method. On a Warp array of ten cells, the image is divided into ten regions, by column, as shown in Figure 3. This gives each cell a tall, narrow region to process; for 512×512 image processing, the region size is 52 columns by 512 rows. To use technical terms from weaving, the Warp cells are the "warp" of the processing; the "weft" is the rows of the image as it passes through the Warp array.

The image is divided in this way using a series of macros called GETROW, PUTROW, and COMPUTEROW. GETROW generates code that takes a row of an image from the external host, and distributes one-tenth of it to each of ten cells. The programmer includes a GETROW macro at the point in his program where he wants to obtain a row of the image; after the execution of the macro, a buffer in the internal cell memory has the data from the image row.

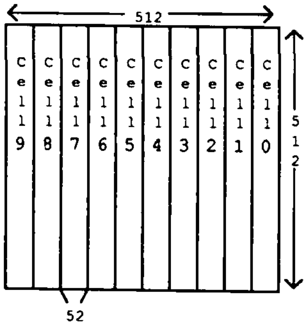


Figure 3: Input Partitioning Method on Warp

The GETROW macro works as follows. The external host sends in the image rows as a packed array of bytes – for a 512-byte wide image, this array consists of 128 32-bit words. These words are unpacked and converted to floating point numbers in the interface unit. The 512 32-bit floating point numbers resulting from this operation are fed in sequence to the first cell of the Warp array. This cell takes one-tenth of the numbers, removing them from the stream, and passes through the rest to the next cell. The first cell then adds a number of zeroes to replace the data it has removed, so that the number of data received and sent are equal.

This process is repeated in each cell. In this way, each cell obtains one-tenth of the data from a row of the image. As the program is executed, and the process is repeated for all rows of the image, each cell sees an adjacent set of columns of the image, as shown in Figure 3.

We have omitted certain details of GETROW – for example, usually the image row size is not an exact multiple of ten. In this case, the GETROW macro pads the row equally on both sides by having the interface unit generate an appropriate number of zeroes on either side of the image row.

Also, usually the area of the image each cell must see to generate its outputs overlaps with the next cell's area. In this case, the cell copies some of the data it receives to the next cell. All this code is automatically generated by GETROW.

PUTROW, the corresponding macro for output, takes a buffer of one-tenth of the row length from each cell and combines them by concatenation. The output row starts as a buffer of 512 zeroes generated by the interface unit. The first cell discards the first one-tenth of these and adds its own data to the end. The second cell does the same, adding its data after the first. When the buffer leaves the last cell, all the zeroes have been discarded and the first cell's data has reached the beginning of the buffer. The interface unit then converts the floating point numbers in the buffer to zeroes and outputs it to the external host, which receives an array of 512 bytes packed into 128 32-bit words. As with GETROW, PUTROW handles image buffers that are not multiples of ten, this time by discarding data on both sides of the buffer before the buffer is sent to the interface unit by the last cell.

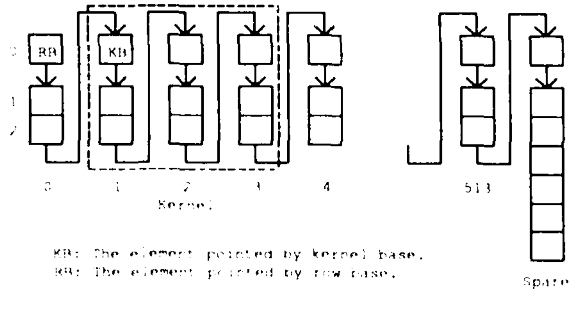
During GETROW, no computation is performed; the same applies to PUTROW. Warp's horizontal microword, however, allows input, computation, and output at the same time. COMPUTEROW implements this. Ignoring the complications mentioned above, COMPUTEROW consists of three loops. In the first loop, the data for the cell is read into a memory buffer from the previous cell, as in GETROW, and at the same time the first one-tenth of the output buffer is discarded, as in PUTROW. In the second loop, nine-tenths of the input row is passed through to the next cell, as in GETROW; at the same time, nine-tenths of the output buffer is passed through, as in PUTROW. This loop is unwound by COMPUTEROW so that for every 9 inputs and outputs passed through, one output of this cell is computed. In the third loop, the outputs computed in the second loop are passed on to the next cell, as in PUTROW.

4.2. Apply on the Sun

The Sun 3/75 workstation is a conventional serial computer with a MC68020 microprocessor, MC68881 coprocessor, and a 16MB main memory. In image processing, all images are stored in the main memory.

The Sun C compiler does not do sophisticated analysis of index expressions to simplify access to arrays. The Apply C compiler employs a special technique called *cyclic-scroll buffering* here, which efficiently uses small space and time to buffer the rows of the image. The technique allows the kernel to be shifted and scrolled over the buffer with very simpler and more efficient index expressions than would be generated if the image was stored as a large array.

The cyclic-scroll buffering technique which we developed for Apply on *uni-processor machines* is described as follows. For an $N \times N$ input image which will be processed with an $M \times M$ kernel, a buffer with $(N+M-1) \times M + (N-1)$ elements is required.



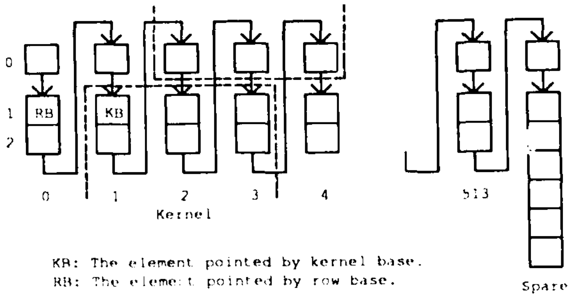
0
1
2
RB
KB
0
1
2
3
4
513
Kernel
KB: The element pointed by kernel base.
RB: The element pointed by row base.
Spare

Figure 4: Processing the first row by the cyclic-scroll buffering

Figure 4 and 5 display the column-major arrangement for processing a 3×3 kernel. The pointers represent successive positions in memory. In addition, we keep two base pointers for the buffer. One, called *row base*, points to the first pixel of the three rows of the image and the other, called *kernel base*, points to the first pixel of the kernel. C language subscripting can be used to directly access the elements of the kernel except that the indices of row and column must be exchanged because the rows of the images are stored in column-major order.

Initially, we put the first M rows of the image, including the border, into the buffer in *column-major order*. When the first kernel is processed, row base points to the first element of the buffer, and kernel base points to the first element of the window to be processed. After the first kernel has been processed, the kernel base is incremented by M to point to the first pixel of the next kernel. It is thus possible to shift the kernel across the entire buffer of data with a cost of only one addition.

When processing an entire row is completed, the first row in the buffer from the row base is discarded and the next row of the image is input into the discarded row with a column displacement of one (i.e., beginning at the second element). Then the row base is incremented by one. The purpose of column displacement 1 is that the input row can be considered to be the M^{th} row of the buffer starting from the new row base. Effectively, the rolling is done at the same time. After the kernel base is reset to point to the center element of the new window, we can do another row operation in the same way as the first until all the rows are processed. Figure 4 and 5 show the processing of the first and second row.



0
1
2
RB
KB
0
1
2
3
4
513
Kernel
KB: The element pointed by kernel base.
RB: The element pointed by row base.
Spare

Figure 5: Processing the second row by the cyclic-scroll buffering

For each row operation, one more memory element is needed in the buffer Therefore, the total number of the elements in the buffer is supposed to be $M \times (N+M-1) + (N-1)$.

4.3. Apply on the Hughes HBA

The Hughes HBA is a 24 processor MIMD computer specifically designed for image processing applications. The main architectural novelty of the HBA is its video i/o bus, a digital bus that can broadcast copies of a digitized image to all 24 processors from an external buffer in one frame time. Each processor (called an IP – image processor) supports vision programming with a general purpose CPU (MC68020 with a 12 MHz clock rate at the time of this study), floating-point coprocessor (MC 68881) and ample memory (1 MB). The IPs are also linked by a low-bandwidth (80 MHz) communications bus intended for message passing among the IPs and the host computer. Software support for all levels of image processing includes a high-level (C) language compiler, debugger, and an operating system (HOPS) supporting video data transfers, message passing, memory allocation, tracing and downloading software from the host. In support of low-level pixel processing operations the HBA runs the Apply routine.

The HBA implementation of Apply uses the video I/O bus to transfer image data from a frame buffer to the IPs. Although it is possible to transfer the entire image to each of the IPs, it is only necessary for each IP to look at a part of the image to achieve linear speedup of the Apply operation. An IP receives data from the video bus by specifying a starting scanline and a number of scanlines to be received. The data on the video bus appears in scan-sequential order so the IP opens a DMA pathway to its local memory when its starting scanline appears on the bus. An IP outputs video data in much the same way, by setting the appropriate control registers with the start scanline, number of scanlines to output, and location of the video data in its local memory. The set of consecutive scanlines an IP receives or transmits is called a swath.

For the Apply operation, an input swath is always at least as large as an output swath. If the input image is size R rows by C columns, and the programmer wants to compute an output image with an Apply operator using N processors, each IP computes a swath of $S=floor(R/N)$ rows by C columns. If p is a processor number in the range $0..N-1$, then each output swath extends from row Sp to row $S(p+1)-1$. If the Apply operation is defined over a window that extends from $[lbr,lbc]$ to $[ubr,ubc]$, then each processor p must input a swath that extends from $Start=Sp+lbr$ to $End=S(p+1)+ubr-1$. If $Start<0$ the input routine inserts $-Start$ rows of 0's in place of the video data and advances $Start$ to 0. The case of $End>R-1$ is similar.

Internal to each IP the Apply operation is implemented using an Illiffe vector method [3]. In the HBA, a scanline of pixels occupies connective memory locations. The Apply implementation uses an Illiffe vector of $ubr-lbr+1$ row pointers to hold address offsets into scanlines. To access a pixel at position $[r,c]$ relative to the coordinate frame of the window the Apply routine finds the value in the memory address offset by c from the r^{th} row pointer. Each time the Apply program computes an output value for a window, the column vector advances to the next column of pixels. When the window reaches the end of a scanline, Apply shifts the row pointers up and sets the bottom value to the starting address of the next scanlines to be processed.

5. Apply Code Compared with Hand-written Code

Our primary purpose in this paper is to develop a comparison of different parallel processing machines for vision using Apply as a vehicle. In order to base this comparison on solid ground, we must first evaluate Apply's performance compared with hand-written code for the same machine. If Apply produces code that is comparable to hand-written code, then our comparison will be solidly based, since the code generated by Apply represents the peak performance of the machine. On the other hand, if Apply code is not as good, then the comparison will not be solidly based; it could be argued that the measured performance would not actually be seen, since the user would not use Apply.

5.1. Apply code compared with SPIDER code

We begin by comparing Apply performance on WEB routines with a set of routines of similar function from the SPIDER FORTRAN library. The SPIDER library is professionally written and distributed, and the code is of high quality; therefore, this comparison pits Apply's code against the code of expert programmers.

We are comparing the actual execution times (user time plus system time) of the FORTRAN programs, called as a subroutine from C, with execution times of C programs generated by Apply, called in the same way. The time is measured from the point at which the input images are ready (have been stored in the Sun's memory) to the point at which the output images are ready, in both cases. This time does not include the I/O time for the images from disk, or the code download time from disk into the Sun. All times are for 512×512 images.

Figure 6 gives the ratios of execution times for these programs, and Figure 7 shows the distribution of times for all programs. We can see from these figures the following phenomena:

- The Apply programs are generally faster. There are four factors that can account for this: (1) Cyclic-scroll buffering; (2) The superiority of the Sun C compiler to the Sun FORTRAN compiler; (3) The FORTRAN code is written to be readable, at the expense of efficiency; the code generated by Apply need not satisfy such a constraint, since the Apply input code is quite readable. Apply can sacrifice legibility for speed.

- In some cases such as addplr and divclr the Apply code is slower. In these programs the algorithm is processing a single pixel from the input image to produce a single pixel in the output image. The cyclic-scroll buffering technique introduces a significant overhead in this case. (The same does not apply for addplb and addclb since here the FORTRAN code is processing integer images, while the Apply program is processing byte images. Thus, these programs are not strictly comparable).

- Apply has some limitations in its programming model that affect performance. In the FORTRAN subroutines, it is common to write several different ways of computing the output depending on switches. There is little overhead for this in FORTRAN since the code can be written so as to test the value of the switches once per image. In Apply, the equivalent code would test the value of the switch once per pixel, since the Apply procedure is executed in its entirety for every pixel. This can limit

5.2. Apply code compared with W2 code

Next we compare performance on the Carnegie Mellon Warp machine. This machine is programmed by hand in W2, a Pascal-level language in which the user is explicitly aware of the different processors and the communication between them. (Send and receive statements are used to send words of data between cells).

While many programs have been and continue to be written for Warp in W2, the availability of Apply has significantly eased the programmer's burden. Apply hides the explicit parallelism of cells, the number of cells in use, and the communications between cells from the programmer. This has made it possible to develop WEB for Warp; without Apply, it is doubtful that such a library could have been built.

Figures 8 and 9 give the performance for hand-written W2 programs compared with WEB programs of equivalent function. The times are measured from the moment the input data is available for processing in the external host memory by the Warp array to the moment the output data is stored into the external host memory. All times are for 512×512 images. There are three main phenomena responsible for the wide distribution of execution time ratios:

- Some programs, such as egrs1, egpw3, and egpw4, are much slower in W2 than in Apply. This is because the W2 programmer made less optimizations to the code (such as unrolling innermost loops) than the Apply programmer. The Apply programs are smaller, and do not include statements for I/O, so that the programmer's effort is focused on making the heart of his code as efficient as possible.

- The Apply-generated programs consistently overlap I/O with computation on the cell, while the W2 programs do not. This is because, while it is possible to communicate between cells in the same Warp microinstruction where computation is done, doing so involves some careful placement of I/O statements which can be hard for the programmer.

- Some programs are slightly faster in W2 than in Apply. This is because of the limitation of the Apply programming model discussed earlier; the W2 programmer can initialize state based on the values of global variables, and avoid retesting switches, etc., during processing of the images, while the Apply programmer cannot do this.

Here we see two principal effects of the Apply language on Warp programming: (1) The Apply programs are simpler and easier to write, so the programmer makes them more efficient, and Apply in turn generates better code because it can deal with the machine complexity better; (2) The limitation of the Apply programming model for preprocessing data can lead to some loss of performance.

6. Comparison of Diverse Architectures

It is very rare that widely different computer architectures are compared directly for performance; the best previous examples have been FORTRAN studies of supercomputer performance [4]. These have depended on the implementation of a large language designed for use on sequential computers, and so have been limited to those computers in which significant software development has occurred to bring up FORTRAN, and which are suitable for implementation of a sequential computer language.

The comparisons presented here differ from these because the Apply language is designed for use on parallel computers, so that a wider range of computers can be compared, and because Apply is application-specific, so that it is small and does not require an enormous effort to bring up on a new system. Thus, we are able to directly compare the Sun 3/75, the Carnegie Mellon Warp machine, and the Hughes HBA.

6.1. Warp Compared with Sun

Figures 10 and 11 give the performance of a large number of programs implemented both on the Sun 3/75 computer and the Warp machine. This allows us to evaluate the Warp's performance for image processing compared with a high-performance workstation.

Warp execution time was measured from the point at which the arrays of input data were available in Warp's external host to the point at which the arrays of output data became available. This is consistent with the measurement method for the Sun 3/75. Code download time was not included. All times are for 512×512 images.

We observe the following from these data:

- There are a few cases where Warp's performance far exceeds expectations: ir. the case of egfc and egks2, for example. Based on the comparative floating point rates, we would expect Warp's performance to be one to two hundred times that of the Sun, but here the execution times is 666 and 304 times less than the Sun. These large factors are due to the internal parallelism of the Warp cell; it consists of many independent units, which can be individually controlled with a wide horizontal microinstruction. In the best case, a Warp cell can do I/O with other cells, read and write memory, compute an integer ALU operation, and compute a floating point add and a floating point multiply, all in the same 200 nanosecond cycle. The success of this design (and of the compiler in packing instructions together) is shown in the ratios for egfc and egks2.
- In the majority of cases, the execution time ratio is tens of times the Sun 3/75. (The average ratio is 67, with a median of 40). This reflects the raw processing power of Warp combined with the effects of the applications mix (which includes a large amount of integer processing) and the efficiency of the Warp compiler.
- In some cases, the ratio is ten or less. In these cases, the Apply program cannot make use of Warp's highly pipelined floating point units, because of a large amount of conditional branching within the program, and also because the computation is mainly additions, so that the separate multiplier cannot be used. Here we are seeing the effects of using a highly pipelined machine to implement what is essentially a scalar operation. The multiple independent Warp cells can still be used effectively, since the computation is independent between each cell; but the pipelining of the computation within the cell is not successful.

6.2. Warp Compared with Hughes HBA

The Hughes HBA and Warp satisfy very different applications requirements. One is a machine specifically designed for image processing, with a special video interface, and all high-speed I/O through a frame buffer; the other is a machine interfaced to a general-purpose external host, which is suited for scientific computing and signal processing as well as image processing. The high-speed floating point in Warp is largely a reflection of the desire to satisfy all of these applications areas.

The HBA times are measured from the time the image is available for processing in the frame buffer of the HBA to the time the output image is stored there. At the time of this study, the HBA processed 240×256 images; to be consistent with the Warp times, the HBA times have been multiplied by 4.27. The Warp times are for 512×512 images, measured as before.

We have done only preliminary work on the comparison between the Hughes HBA and Warp. Only a few programs have been tested, and most of those are integer applications, biasing the data against Warp, since it has higher-speed floating point than the HBA. Taking this into account, we can study the data shown in Figure 12 and 13:

- The Warp times are, on average, 3.2 times better than the HBA (the median is also 3.2). This reflects the greater total computational power of the Warp compared with the HBA, together with the application bias towards conditional, integer applications – the HBA can execute approximately 25 MFLOPS versus the Warp's 100 MFLOPS.
- In the fmin and fmax algorithms, which compute the minimum and maximum of a 3×3 window around each pixel, the HBA time is slightly better than the HBA time. This is because in such a highly conditional operation, the use of the long pipeline inside the Warp cell is a barrier to good performance. Moreover, the total number of ALU operations in the HBA is greater, since there are 24 processors instead of 10, of comparable integer performance.

7. Conclusions

This is the first study which we are aware of in which highly diverse architectures have been compared using the same source code. We can make several conclusions based on this study:

- Apply and WEB are clearly good tools for comparing these architectures. Quite apart from the utility of having Apply and WEB available on an architecture, which is considerable, using the same source code eliminates many factors that significantly affect performance but which are irrelevant to performance analysis. Apply is easy to implement on a parallel processor, which makes it possible to evaluate the performance of a large number of parallel machines with little effort. We look forward to evaluating other parallel architectures in the future.
- The main performance limitation of the Apply programming model is the inability to manipulate constant-type parameters once per image rather than once per pixel. This deficiency will have to be corrected in future versions of Apply or its successors.
- Most of the performance limitations of one architecture over another arise from intra-processor characteristics, rather than inter-processor characteristics. This is because this level of vision is easy to parallelize, so that different processors need to communicate very little. Much more significant is the ability of the processor to successfully implement the wide range of operations that is required in low-level vision, including integer, floating point, and conditional operations.
- Parallel processors deliver performance increases even over high-performance workstations at this level of vision, and Apply makes them no harder to program. The performance ratios vary from ten- to hundred-fold increases. This is a significant, cost-effective, performance increase.

Acknowledgments The original idea for Apply came from discussions between Steve Shafer and Leonard Hamey of Carnegie Mellon. Leonard Hamey designed and implemented the first Apply compiler, and consulted on the current implementation. Hudson Ribas of Carnegie Mellon implemented most of the WEB library. Truongthinh Le of Carnegie Mellon developed the data for evaluating Apply's performance on the Sun. The research was supported in part by Defense Advanced Research Projects Agency (DOD), monitored by the Air Force Avionics Laboratory under Contract F33615-81-K-1539, and Naval Electronic Systems Command under Contract N00039-85-C-0134, in part by the US Army Engineer Topographic Laboratories under Contract DACA76-85-C-0002, and in part by the Office of Naval Research under Contracts N00014-80-C-0236, NR 048-659, and N00014-85-K-0152, NR SDRJ-007.

References

[1] Annaratone, M., Arnould, E., Gross, T., Kung, H. T., Lam, M., Menzilcioglu, O. and Webb, J. A.
The Warp Computer: Architecture, Implementation and Performance.
IEEE Transactions on Computers C-36(12), December, 1987.

[2] Hamey, L. G. C., Webb, J. A., and Wu, I-C.
Low-level Vision on Warp and the Apply Programming Model
Parallel Computation and Computers for Artificial Intelligence.
Kluwer Academic Publishers, 1987.
Edited by Janusz Kowalik.

[3] Illiffe, J. K., and Jodeit, J. G.
A dynamic storage allocation scheme.
The Computer Journal 5:200-209, 1962.

[4] Jordan, Kirk E.
Performance comparison of large-scale scientific computers: Scalar mainframes, mainframes with integrated vector facilities, and supercomputers.
IEEE Computer 20(3):10-23, March, 1987.

[5] Electrotechnical Laboratory.
SPIDER (Subroutine Package for Image Data Enhancement and Recognition).
Joint System Development Corp., Tokyo, Japan, 1983.

[6] Wallace, R. S. and M. D. Howard.
HBA Vision Architecture: Built and Benchmarked.
In *Computer Architecures for Pattern Analysis and Machine Intelligence.* IEEE Computer Society, Seattle, Washington, December, 1987.

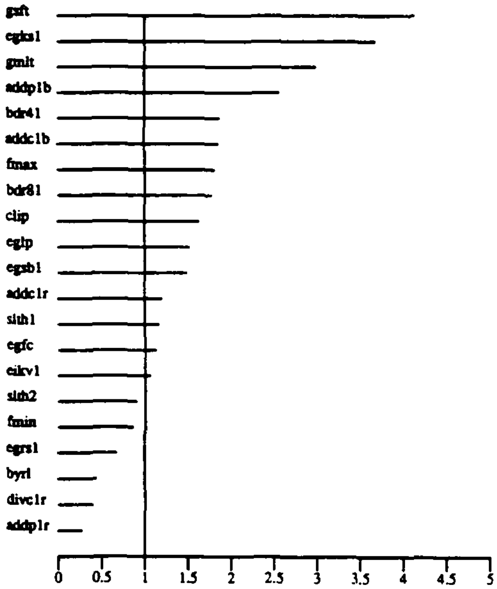


Figure 6: Ratio of execution times of hand-generated SPIDER FORTRAN to Apply code.
Vertical line indicates a ratio of one.

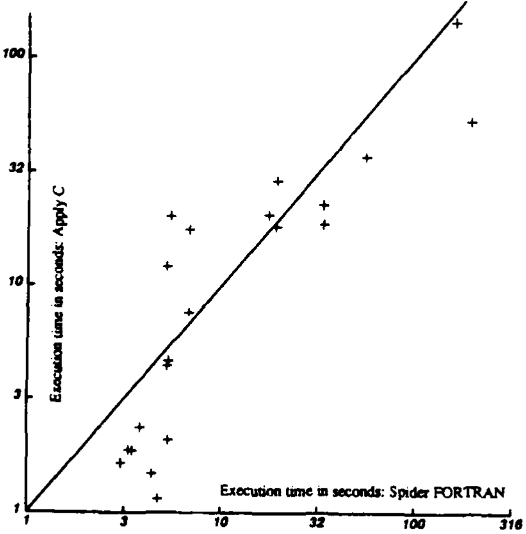


Figure 7: Scatter diagram of execution times of hand-generated SPIDER FORTRAN and Apply code.
Diagonal line indicates equality.

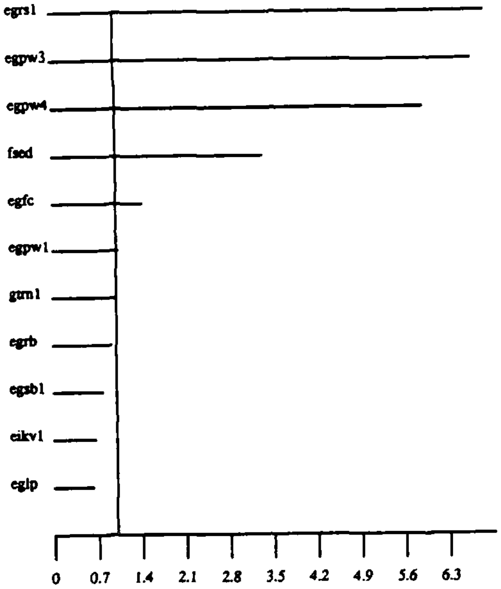


Figure 8: Ratio of execution times of hand-generated W2 code to Apply code.
Vertical line indicates a ratio of one.

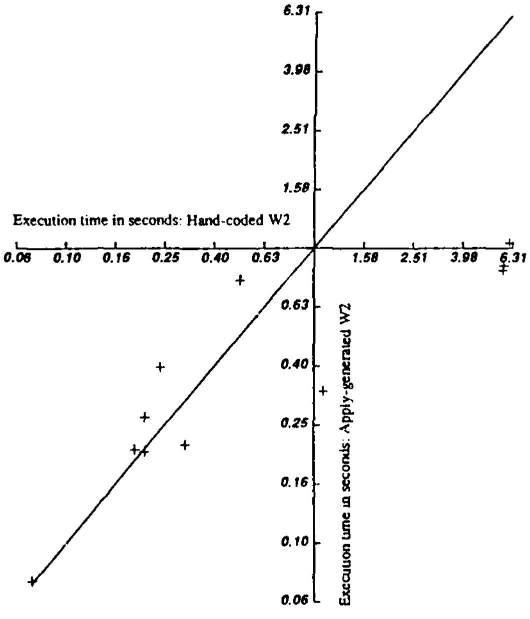


Figure 9: Scatter diagram of execution times of hand-generated W2 code and Apply code.
Diagonal line indicates equality.

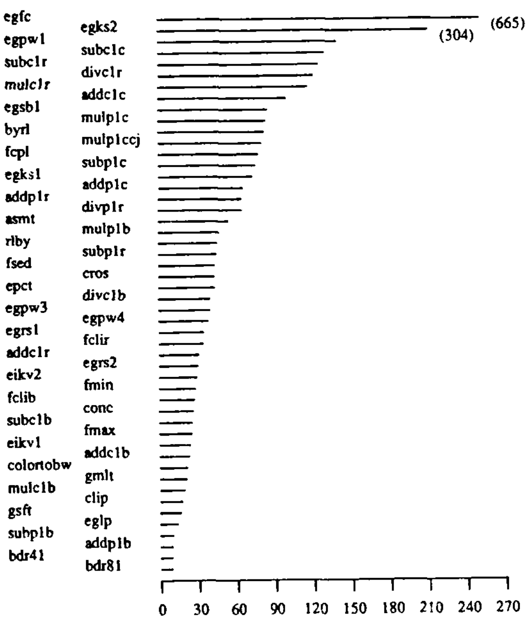


Figure 10: Ratio of execution times of Sun Apply code to Warp Apply code.

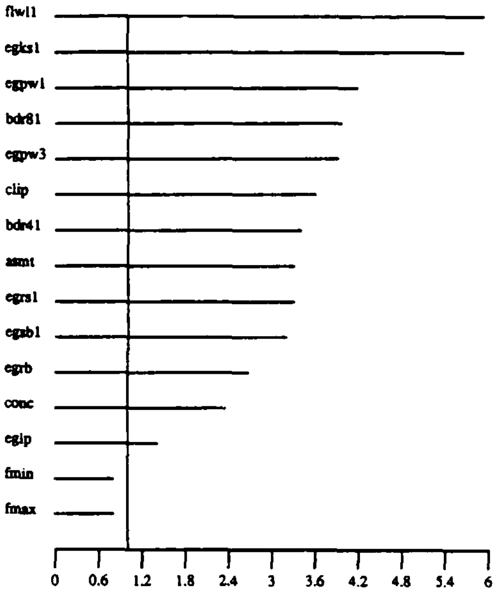


Figure 12: Ratio of execution times of Hughes HBA Apply code to Warp Apply code. Vertical line indicates a ratio of one.

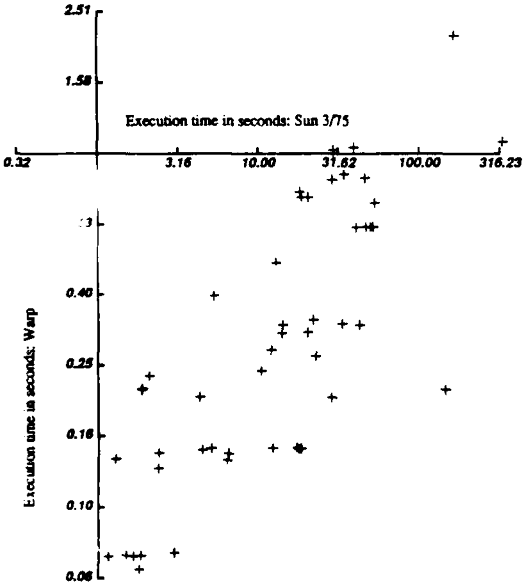


Figure 11: Scatter diagram of execution times of Sun Apply code and Warp Apply code.

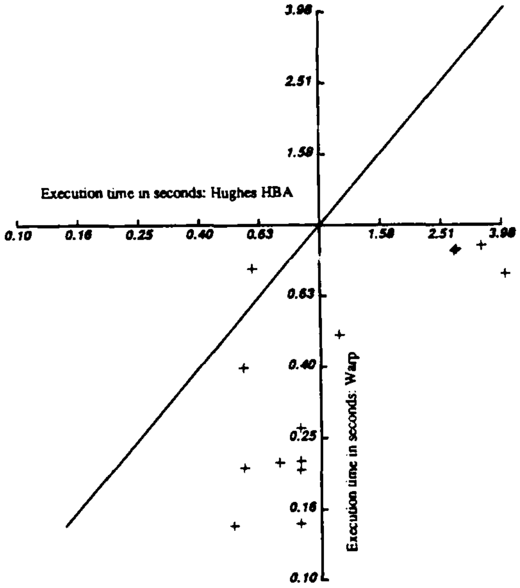


Figure 13: Scatter diagram of execution times of Hughes HBA Apply code and Warp Apply code. Diagonal line indicates equality.

Parallel Architectures for Image Processing and Vision [1]

V. K. Prasanna-Kumar and Dionisios Reisis
Institute for Robotics and Intelligent Systems
University of Southern California
Los Angeles, CA 90089-0273

Abstract

In this paper, we summarize the activities at USC in developing parallel architectures and parallel techniques for various image and vision problems. The proposed architectures include the enhanced mesh, the reconfigurable mesh and the mesh of meshes. These architectures are illustrated by parallelizing known techniques and/or developing new parallel algorithms for image processing and vision. We develop efficient algorithms to support data routing among the processors. By using these routing techniques, we solve a variety of problems related to digitized images, as well as problems in image understanding using linear segments as primitives.

1 Introduction

Image processing and vision tasks are usually computationally intensive. A variety of techniques have been developed for efficient solutions to such problems, from low level processing of digitized images, to high level processing, as in image understanding. Many parallel architectures have been proposed to efficiently implement these techniques ([12], [13], [7],[14], [15], [57], [64], [32], [33], [54], [49], [47], [4], [62]).

In this paper, we summarize the activities at USC in developing parallel architectures and parallel techniques for various image and vision problems. The proposed architectures include the enhanced mesh, the reconfigurable mesh and the mesh of meshes. These architectures are illustrated by parallelizing known techniques and/or developing new parallel algorithms for image processing and vision. We develop efficient algorithms to support data routing among the processors. By using these routing techniques, we solve a variety of problems related to digitized images, as well as problems in image understanding using linear segments as primitives.

In the next section, we address architectures and algorithms for information extraction from digitized images. The 2-dimensional mesh connected computer is a well studied parallel architecture for solving such problems [51,32]. The mesh organization, from a hardware point of view, is superior to other models with respect to modularity, regularity and its low interconnection cost. A prohibiting factor in achieving large speedups using the mesh is its diameter, which is $\Omega(N^{1/2})$ for a $N^{1/2} \times N^{1/2}$ mesh. The proposed models include modifications, which alleviate the $\Omega(N^{1/2})$ running time performance of the mesh. For example, the addition of broadcast buses to the mesh results in the *enhanced mesh* having time performance comparable to that of a pyramid for many image problems. We illustrate elegant parallel solutions on the mesh of trees, an organization not well studied by researchers in image processing and vision. We also show a related architecture, the *mesh of meshes*. On this model we can achieve linear speedup in solving several image problems, while the number of processors can vary over a wide range. A more powerful architecture, the *reconfigurable mesh*, seems to be a universal array for image computations, providing the ability to simulate both the pyramid and the mesh of trees organization. Moreover, on the reconfigurable mesh, superior asymptotic time performance can be obtained, compared to the best known performance on the pyramid and the mesh of trees organizations.

Over the past decade, several techniques have been developed at USC for problems in image understanding. Among these, using linear segments as primitives [29,30] has been well studied. In section III of the paper we consider parallelizing such techniques. We develop an input partitioning strategy for the parallel solution of the *image matching* and the *stereo matching* problems using linear segments as input. This leads to a parallel implementa-

[1] This work was supported in part by the National Science Foundation under grant IRI-8710863 and in part by DARPA under contract F 33615-87-C-1436, order No 3119, monitored by the Air Force Wright Aeronautical Laboratory.

tion of such algorithms achieving linear speedup on a mesh connected computer. We also present a systolic implementation of these algorithms. This achieves a speedup of $O(k^{1/2})$ on a systolic array with $k + k^{1/2}$ cells, $4 \leq k \leq m$, where m is the number of linear segments in the image.

Details of the work presented here can be found in [4,44,47,48,49]. We thank Hussein Alnuweiri and Mary Eshaghian, for sharing their contributions in this paper.

2 Parallel Architectures and Algorithms for Image Computations

In this section we present parallel architectures and algorithms for solutions to problems related to digitized images. The input to these algorithms is a digitized image of size $N^{1/2} \times N^{1/2}$. Connected 1's through neighbor to neighbor connections are called figures [50].

In all our analysis we make the following assumptions: each processor can perform arithmetic and logic operations in $O(1)$ time. Each processor has a constant number of registers serving as local memory. A register can hold $O(\log N)$-bit data. A packet (record), which consists of $O(\log N)$ bits, can be transmitted using a connection link between two processors in $O(1)$ time.

2.1 Enhanced Mesh

The enhanced mesh of size $N^{1/2} \times N^{1/2}$ is based on a 2-dimensional mesh connected computer (2-MCC) of size $N^{1/2} \times N^{1/2}$. Each processor is placed at grid points (i, j), $0 \leq i, j \leq N^{1/2} - 1$. In addition to the standard 2-MCC organization we allow broadcast in each row and each column. In other words, each processor in a row (column) has a port connected to the corresponding row (column) broadcast bus. During a broadcast cycle, among the $N^{1/2}$ processors sharing a bus, at most one processor can write onto the bus. All the processors sharing a bus can read from the bus. An enhanced mesh of size 4 × 4 is shown in figure 1. Using the broadcast bus a processor can transmit a record to another processor in the same row (column) in $O(1)$ time. Notice that such a scheme reduces the time needed for communication between any two processors to $O(1)$. A clever use of the broadcast buses is of significant assistance in developing fast operations among data [43]. A set of such operations which makes the enhanced mesh an attractive scheme for implementing image understanding algorithms is:

1. Any associative binary operation among N data items can be performed in $O(N^{1/6})$ time.

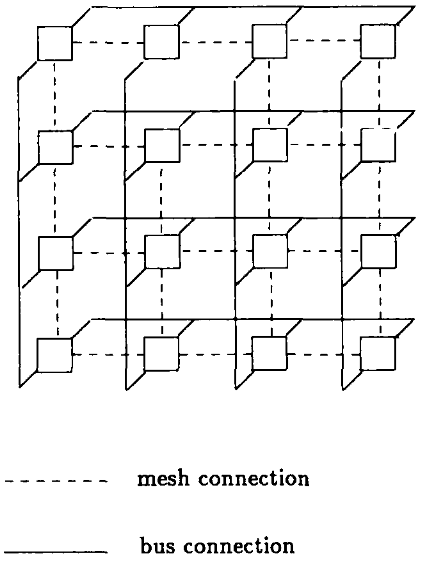


Figure 1: Enhanced mesh

2. $N^{1/2}$ data stored one item per processor in a row (column) can be sorted in $O(\log^2 N)$ time and an associative binary operation on these data can be performed in $O(\log N)$ time.

Data routing can be accomplished by the following operations developed for SIMD machines [37]: the *Random Access Read (RAR)* and *Random Access Write (RAW)*. These operations can be informally described as follows: each processor $PE_{i,j}$, $0 \leq i, j \leq N^{1/2} - 1$, has a data $D(i, j)$ and an address $A_{i,j}$. The address corresponds to the index of a processor in the array. In RAR, $PE_{i,j}$ is to read the data $D(A_{i,j})$. In RAW, each $PE_{i,j}$ sends its data to the processor pointed to by its address register (details can be found in [37]). These operations take $O(N^{1/2})$ time on the 2-MCC as well as on the enhanced mesh. In cases where the distribution of source and destination processors is *sparse*, the broadcast buses can support significantly faster data movement. Such sparse data movement is often needed in parallel solutions to image problems. *Sparsity conditions* that allow fast data movement using the buses are:

1. In any block of size $k \times k$, $1 \leq k \leq N^{1/2} - 1$, the number of source and destination processors is $\leq k$. Then, random access read and write can be performed in $O(N^{1/6})$ time on the enhanced mesh [47].

2. In any block of size $k \times k$, $1 \leq k \leq N^{1/4}$, the number of source and destination processors is $\leq k$. In this case, random access read and write can be performed in $O(N^{1/4})$ time [47].

Algorithms for processing digitized images on the enhanced mesh are asymptotically much faster compared to those on the mesh. In many instances the performance of the algorithms on the enhanced mesh is comparable to those on the pyramid computer of the same size [47]. We outline an algorithm for labeling all the figures in an image, as an illustration of the use of broadcast buses.

The idea here, is to divide the entire mesh into blocks of size $k \times k$ and label the parts of the figures which are lying inside each block for all the blocks in parallel. Blocks are merged in parallel to yield new labels in larger size blocks, by using the connectivity information along the boundary of adjacent blocks. Figures in adjacent blocks are merged as follows:

Assume that the MCC is divided into blocks of size $k \times k$ and all the figures inside each block are labeled. To relabel the figures of two neighboring blocks we use the following representation. Each label corresponds to a vertex. Let p and q be nodes of the graph with corresponding labels l_p and l_q. Suppose we are merging two blocks horizontally. Then, an edge between the nodes p and q exists in the graph, if and only if there exists a pair $PE_{i,j}$ and $PE_{i,j+1}$ at the boundary between two blocks, such that $PE_{i,j}$ has label l_p and $PE_{i,j+1}$ has label l_q. To compute the connectivity of the figures belonging to two adjacent blocks, we examine the connected components of the graph defined above. The input to the algorithm during each iteration is a set of m ($m \leq k$) edges for each pair of adjacent blocks of size $k \times k$. The input data set is stored in a linear array formed by the border processors of the two blocks to be merged. The size of the array is k. Blocks are merged in three phases as follows [47]:

In the first phase, we merge blocks using the mesh connections, until the block size is $N^{1/4} \times N^{1/4}$. This takes $O(N^{1/4})$ time.

We implement the second phase as follows: In each block of size $l \times l$, we move the l edges of the graph in a block of size $\sqrt{l} \times \sqrt{l}$. Four adjacent blocks of size $l \times l$ can be merged in $O(\sqrt{l}\log l)$ time. During the ith iteration of the second phase, merging four blocks of size $l \times l$, where $N^{1/4} \leq l \leq (N^{1/2}/\log^2 N)$, takes $O(2^i N^{1/8} \log N)$ time for the connected components algorithm and $O(N^{1/4}/2^i)$ time for communication on the mesh by using the broadcast buses, $0 \leq i \leq \log(N^{1/8}/\log^2 N)$. The entire second phase takes $O(N^{1/4})$ time.

The last phase of the algorithm starts when the size of the block is $(N^{1/4}/\log^4 N) \times (N^{1/4}/\log^4 N)$. Now, there are $N^{1/2}/\log^8 N$ buses available to each block. There are $2N^{1/2}/\log^4 N$ edges (in each block) to be used for the connected components algorithm. Using a well known technique [21] for computing connected components, the last merging phase can be completed in $O(\log^9 N \log\log N)$ time using the broadcast buses. This leads to:

Theorem 2.1 *On an enhanced mesh of size $N^{1/2} \times N^{1/2}$, given a $N^{1/2} \times N^{1/2}$ digitized image, all figures can be labeled in $O(N^{1/4})$ time.*

Notice that the labeling problem takes $O(N^{1/4})$ time on a pyramid computer of base $N^{1/2} \times N^{1/2}$ [31].

Using the buses to perform sparse data movement, a closest figure to each figure can be computed [47]:

Theorem 2.2 *On an enhanced mesh of size $N^{1/2} \times N^{1/2}$, given a $N^{1/2} \times N^{1/2}$ digitized image, a nearest figure to each figure can be computed in $O(N^{1/4})$ time.*

The above computation takes $O(N^{1/4})$ time on a pyramid of base $N^{1/2} \times N^{1/2}$ [31]. Buses can be used to efficiently compute geometric properties of images. Using the sparse data movement techniques, it is possible to show [47]:

Theorem 2.3 *On an enhanced mesh of size $N^{1/2} \times N^{1/2}$, the geometric properties of the convex hull of a single figure (such as its diameter, a smallest enclosing box and a smallest enclosing circle) can be computed in $O(N^{1/6})$ time.*

As a comparison, notice that the computation of the above geometric properties takes $O(N^{1/6})$ time on a pyramid of base $N^{1/2} \times N^{1/2}$ [31]. Further on image computations on the enhanced mesh can be found in [47].

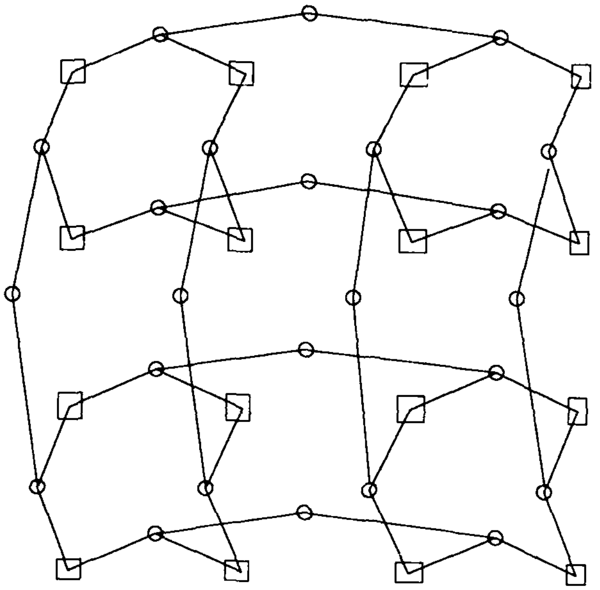
2.2 Mesh of Trees Organization

In the *Mesh Of Trees* organization (MOT) a processor is placed at grid points of a $N^{1/2} \times N^{1/2}$ mesh. Each row and each column of processors forms the leaves of a complete binary tree [27,38]. The root and the internal nodes of each binary tree are also processors. The N leaf processors form the base of the network and they are called *base processors.* A 4×4 MOT is shown in figure 2. In most cases the computations on the mesh of trees are done by the base processors. The upper level processors are used mainly for communication.

The MOT organization has been well studied in the context of VLSI computations [63]. However, not much work has been done in using this organization for image computations.

The MOT can efficiently implement divide and conquer techniques for image problems. As an illustration, we present an algorithm for computing the convex hull of all the figures in an image [44]. The time performance of the algorithm can be further improved. However, in this paper we will illustrate the basic technique.

Assume that the entire image has been divided into disjoint blocks of size $k \times k$. By merging four adjacent blocks of size $k \times k$ we obtain the convex hull of each figure inside blocks of size $2k \times 2k$. This is done for all the figures in parallel. Convex hulls located in two adjacent blocks are merged if they belong to the same figure. Merging can be accomplished by detecting the two common tangent lines to the convex polygons [41]. This is done by performing a binary search on the extreme points of each convex hull. The merging operation of four adjacent blocks takes $O(\log^3 k)$ time, leading to:



□ Leaf Processor

○ Internal Processor

Figure 2: Mesh of Trees Organization

Theorem 2.4 *On a $N^{1/2} \times N^{1/2}$ MOT, the convex hull of all the figures can be computed in $O(\log^4 N)$ time.*

Several geometric properties of a single figure can be easily computed on the MOT. For example:

Theorem 2.5 *The convex hull of a single figure, its diameter, a smallest enclosing box and a smallest enclosing circle of the figure can be computed in $O(\log N)$ time on a $N^{1/2} \times N^{1/2}$ MOT.*

Mesh Of Trees is an efficient parallel organization for problems with sparse information exchange. For example, $N^{1/2}$ data items can be moved from a column to another in $O(\log N)$ time, while such an operation takes $\Omega(N^{1/4-\epsilon})$ time (in the worst case) on a pyramid of base $N^{1/2} \times N^{1/2}$ [31]. Such a feature is very attractive for solving several image problems [44].

2.3 Mesh of Meshes

The *Mesh of Meshes* (MOM) organization has a reduced number of processors compared to other parallel architectures studied for problems on digitized images. In addition, a novel interprocessor communication scheme is supported on the MOM; processors communicate through shared memory locations as well as direct interprocessor links. Using this feature, optimal speed up for a large number of problems in image processing can be obtained [2,3].

The (MOM) organization consists of p processors and n^2 memory locations. The MOM is organized as an array of $k \times k$, $1 \leq k \leq n$, basic modules (BMs). Each BM is a parallel organization consisting of q processors, where $1 \leq q \leq n/k$, and a $q \times q$ array of memory modules. The memory size of each BM is n^2/k^2. Within a BM, each processor has access to one row and one column of the memory modules in the BM. Also, each processor is connected to the four corresponding processors in the four neighboring BMs. Note that the total number of processors in the MOM is $p = k^2 q$, and the total number of memory modules is $k^2 q^2$ (i.e. a total of n^2 memory locations). Such an organization is shown in figure 3 for $q = 4$. The Mesh of Meshes provides optimal speedup for several computations on digitized images when p is in the range 1 to $n^{3/2}$. When $p = n$ the performance of MOM is comparable to the 2-MCC of size $n \times n$ for $n \times n$ image problems. Notice that $\Omega(n)$ is the lower bound on time performance for any *non trivial* problem on the 2-MCC [36]. Most of the image problems can be solved in $O(n)$ time on the MOM when $p = n$.

An interesting result is that on the MOM, n^2 elements can be sorted in $O(n^2 \log n/p)$ time, for $1 \leq p \leq n \log n$ [3].

Scanning through values stored in a memory row or in a memory column can be performed in $O(n/k + k)$ time. Suppose, a $n \times n$ digitized image is stored on the mesh of meshes, one pixel per memory location. The convex hull of a set of 1's can be computed in two steps. First, the leftmost and the rightmost 1's on each row are identified. Second, for each of the above detected points we compute the angles with all the other points, to decide if it is an extreme point. These two steps can be performed, by scan operations.

Using scan of data and efficient data reduction and data movement, we can show [2]:

Theorem 2.6 *The mesh of meshes organization with p processors can perform the following computations on a $n \times n$ image in $O(n^2/p)$ time, $1 \leq p \leq n^{3/2}$:*

1. *Labeling all the figures in the image.*
2. *Constructing the convex hull of a set of pixels.*
3. *Computing the diameter and a smallest enclosing box (circle) of a set of pixels.*
4. *Computing the closest figure to each figure.*

When $k = n^{1/2}$, the number of processors in the mesh of meshes is $n^{3/2}$ and several image problems can be solved

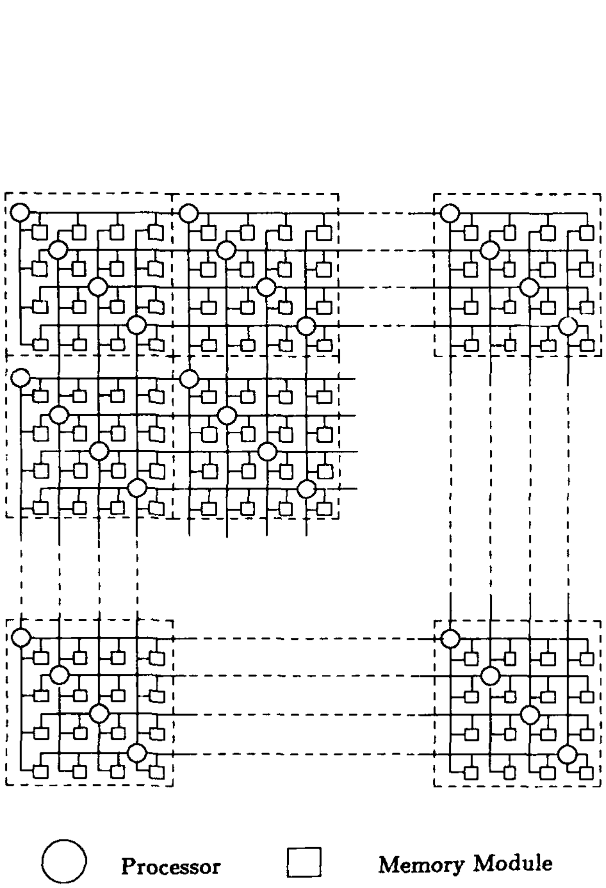


Figure 3: Mesh of Meshes

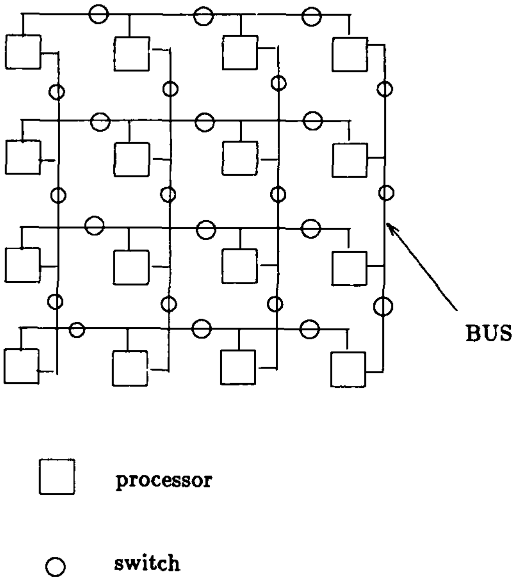


Figure 4: Reconfigurable Mesh

in $O(n^{1/2})$ time. The performance of the mesh of meshes can be compared to the mesh with reduced diameter [33]. The mesh with reduced diameter has $s \times s$ processors with each processor having n^2/s^2 local memory. The mesh with reduced diameter provides linear speedup for several image computations, for s in the range 1 to $n^{2/3}$. All the problems mentioned in the above theorem can be solved in $O(n^2/s^2 + s)$ time. Thus, the fastest possible solution on the mesh with reduced diameter takes $O(n^{2/3})$ time, while it takes $O(n^{1/2})$ time on the mesh of meshes.

Related results and details can be found in [4,2,3].

2.4 Mesh with Reconfigurable Bus

A *mesh with reconfigurable bus (reconfigurable mesh) of size* N consists of a 2-dimensional mesh connected array *of processors (mesh) of size* N, *with each processor con*nected to a broadcast bus. This bus, like the mesh, is also constructed as an $N^{1/2} \times N^{1/2}$ grid, where the processors are connected to the bus at the intersections of the grid. Further, each bus link between intersections has a *switch* embedded in it. The two processors at each end of the link can control the switch. These switches allow the broadcast bus to be divided into subbuses, where each subbus can function as a smaller reconfigurable mesh. Other than the buses and switches the reconfigurable mesh is similar to the standard mesh in that it operates in SIMD (single instruction stream, multiple data stream) mode and has $O(N)$ area, under the assumption that processors, switches, and single links as having constant size.

In each subbus shared by multiple processors, at any given time we assume that at most one processor may use the bus to broadcast a value, where a value consists of $O(\log N)$ bits. Two computational models will be discussed regarding the assumption about the delay that a broadcast requires. The ***unit-time delay model*** will assume that all broadcasts take $\Theta(1)$ time, as is the assumption in [8,43,47,55,56] for models that assume various broadcasting strategies. We will also consider the ***log-time delay model*** in which it is assumed that each broadcast takes $O(\log s)$ time to reach all processors connected to its subbus, where s is the maximum number of switches in a minimum switch path between two processors connected on the bus. In this paper, for sake of simplicity, we illustrate the unit-time delay model. Related results can be found in the joint work with Russ Miller and Quentin Stout who independently investigated a similar organization [35].

Major advantages of the reconfigurable mesh are as follows:

1. Buses can be used to speed up parallel arithmetic and logic operations among data stored in different processors. The reconfiguration scheme supports several techniques on the CRCW PRAM model, leading to the same time performance as in the PRAM model having N processors.

2. The reconfigurable mesh provides an environment for efficient sparse data movement operations.

3. A significant asymptotic improvement can be achieved in the running time of algorithms that solve several problems on the reconfigurable mesh compared to efiicient algorithms for the mesh-of-trees, pyramid and mesh with static broadcast buses.

4. The reconfigurable mesh can act as a universal chip in that VLSI organizations with equivalent area can be simulated without loss in time.

Well known organizations such the Mesh-of-Trees (MOT) and the pyramid computer can be efficiently simulated by the reconfigurable mesh due to the numerous communications patterns that the reconfigurable mesh provides.

A step by step simulation [49] of hierarchical organizations (pyramid, MOT) can be done as follows: Define a c-embedding of a hierarchical organization onto the reconfigurable mesh to have the following properties.

1. A constant number of vertices of the hierarchical organization are mapped to each vertex of the reconfigurable mesh.

2. The number of edges between levels l and $l+1, 0 \leq l \leq k-1$, incident on any row or column bus segment is $\leq c$.

Define a class of algorithms to be *normalized algorithms* if the following hold [63].

1. During a computation step of the algorithm, all data operated on are located at the same level of the hierarchical organization.

2. All communication steps are performed between at most two adjacent levels.

This leads to the following results:

Proposition 2.1

1. Any normalized algorithm running in $T(N)$ time on a mesh of trees of base size N can be simulated on a reconfigurable mesh of size N to finish in $O(T(N))$ time.

2. Any algorithm running in $T(N)$ time on a mesh-of-trees of base size N, can be simulated on a reconfigurable mesh of size N to finish in $O(T(N)\log_2 N)$ time. Further this time is optimal.

3. Any algorithm running in time $T(N)$ on a pyramid of size N can be simulated on a reconfigurable mesh of size N in $O(T(N))$ time.

We now turn our attention to the simulation of the reconfigurable mesh by the pyramid.

Proposition 2.2 *Any algorithm running on the reconfigurable mesh of size N in time $T(N)$ under the unit-time delay model can be simulated on a pyramid of size N in $O(T(N)N^{1/4})$ time. This simulation is optimal.*

Details of the above simulations and the proof of optimality can be found in [49,46].

The reconfigurable mesh can provide efficient communication between processors when the amount of data is sparse. Such fast communication of data is useful in efficient parallel solutions to many problems involving graphs and digitized images.

2.4.1 Image Problems

Many problems involving digitized images can be solved efficiently on the reconfigurable mesh. The input for these problems is an $N^{1/2} \times N^{1/2}$ digitized image distributed one pixel per processor on a reconfigurable mesh of size N so that processor $PE_{i,j}$ has pixel (i,j). The problems that we examine focus on labeling figures and determining properties of the figures.

Theorem 2.7 *Given an $N^{1/2} \times N^{1/2}$ digitized image mapped one pixel per processor onto the processors of a reconfigurable mesh of size N in a natural fashion, the figures can be labeled in $O(\log^2 N)$ time.*

The above theorem is proved by using the divide and conquer approach discussed in section 2.1 and by using the fact, that connected components of a graph represented by its $N^{1/2} \times N^{1/2}$ adjacency matrix can be computed in $O(\log N)$ time on the reconfigurable mesh.

A closest figure to each figure can be computed efficiently on the reconfigurable mesh using the following divide and conquer technique and the reconfigurable buses.

Assuming that the figures are already labeled, the algorithm operates in two phases. In the first phase each processor belonging to a figure, locates the closest processor to itself having a different label as follows. Using the row and the column buses each border processor locates its closest 1's on its row and column using a bus splitting mechanism [46]. By this, the segments of the row and column buses between 1's are isolated. The labels and the indices of the border processors defining each segment are broadcast on the segment. Now, each processor having a 0 and located in between 1's (two 1's in its column and two 1's in its row) computes the closest 1 having a different label. Performing a prefix computation within each segment in parallel for all the segments, each border processor knows which is its closest processor having a 1 with a different label.

During the second phase, bloc' s are merged in a bottom up approach to identify a unique nearest figure to each figure. We start with the assumption that all the processors with the same label located on the periphery of a block of size $k \times k$ have the same information regarding the closest 1 to the label. On the reconfigurable mesh the minimum or maximum of $N^{1/2}$ values, stored in a column can be computed in $O(1)$ time [46]. Using this result, the closest 1 to each figure can be computed within the block of size $2k \times 2k$ in $O(1)$ time. Using the updated information, the processors on the periphery of the $2k \times 2k$ block can start the new iteration. Each iteration of the second phase takes $O(1)$ broadcasts and bus splitting operations leading to:

Theorem 2.8 *Given an $N^{1/2} \times N^{1/2}$ digitized image mappea one pixel per processor onto the processors of a reconfigurable mesh of size N in a natural fashion, in $O(\log N)$ time a closest figure to each figure can be determined.*

In computing the convex hull and the geometric properties of a single figure, efficient use of the reconfigurable bus leads to:

Theorem 2.9 *Using a $N^{1/2} \times N^{1/2}$ reconfigurable mesh, the convex hull of a single figure can be computed in $O(\log N)$ time while its diameter and a smallest enclosing box can be determined in $O(1)$ time.*

Proof of the above theorems as well as related results can be found in [46,49].

3 Parallel processing of segment based input problems

During the past few years, researchers at USC have developed algorithms for image understanding using linear segments as primitives [29,30]. In this section we turn our attention to the parallelization of these algorithms. Here, we illustrate the parallelization of such techniques by studying the *image matching* problem. More details as well as parallel implementation of the *stereo matching* can be found in [48]. First we outline a well known technique for image matching. Then, we present a strategy to partition the input data set. This partitioning technique leads to achieving a linear speedup on a mesh connected computer. We also show a systolic implementation of the algorithm.

Image Matching using Linear Segments

The problem is to match two images. The primitives used by the algorithm are linear *segments* [29]. These are described by their orientation, endpoints of their coordinates and average contrast. The matching procedure is based on the discrete relaxation technique [29]. Let N and M denote the set of vectors in the two images. Let $\mid M \mid= m$, $\mid N \mid= n$ and $n \approx m$. A primitive in the first image is denoted as object a_i, $1 \leq i \leq n$. A primitive in the second image is denoted as label l_j, $1 \leq j \leq m$. The technique computes the quantity $p(i,j)$, in $[0,1]$, which is the *possibility* that object a_i corresponds to label l_j. The assignment of object a_i to label l_j relies on geometrical constraints. This means, that when we assign a label l_j to an object a_i, we expect to find an object a_h with possible assigned label l_k in an area depending on i,j,k. This area is called *window* $w(i,j,k)$. This window is defined as follows: Represent the object a_i by the vector $\vec{A_iB_i}$, the label l_j by the vector $\vec{P_jQ_j}$ and the label l_k by the vector $\vec{P_kQ_k}$. The $w(i,j,k)$ is the parallelogram $R_1R_2S_1S_2$ determined by the equations:

1. $\vec{A_iR_1} = \vec{P_jP_k}$
2. $\vec{R_1S_1} = \vec{P_kQ_k}$
3. $\vec{B_iR_2} = \vec{Q_jP_k}$
4. $\vec{R_2S_2} = \vec{P_kQ_k}$

The pairs (i,j) and (h,k) are called *compatible* $((i,j)C(h,k))$, if a_h is in $w(i,j,k)$ and a_i is in $w(h,k,j)$.

The iteration formula is: For every (i,j), $p^{t+1}(i,j) = 1$, where t denotes the tth iteration, if $p^t(i,j) = 1$ and there exist a subset $S \subseteq M$ such that: $\forall l_s \in S$ there exists a $a_k \in N$ such that $p^t(k,s) = 1$ and $(i,j)C(k,s)$. The algorithm stops when $\forall(i,j)\ p^{t+1}(i,j) = p^t(i,j)$.

3.1 Partitioning Strategy

The proposed architectures have k processing elements PEs $k \leq m,n$. The partitioning strategy is as follows: Initially, when the entire input is loaded to the main memory, detect the vector with the minimum length. Let u denote the length of this vector. Divide the entire image in squares of size $k^{1/2}u \times k^{1/2}u$. In any of these squares there can be at most k vectors. A list of all vectors crossing each region is created. Adjacent squares are merged as long as the total number of vectors in these squares is $\leq k$. During the merge operation, regions are merged so that the number of vectors shared by adjacent regions is minimized. As will be shown in our implementation this partitioning strategy provides linear speed up. This is due the fact that all the k processors of the architecture can be kept busy and the interprocessor communication can be restricted to fetched regions. An example of partitioning an image into regions is shown in figure 3.1.

Notice that the above strategy can lead to increased amount of data to be processed. If the image is sparse this problem does not exist. This happens because although two consecutive square regions contain two versions of the same segment initially, after the merging operation these regions will be merged into one. So finally the segment will be located in a constant number of square regions. In case of a dense image ($m \approx m_{max}$, where m_{max} is the maximum possible number of segments in the image) there can be $(2/3)\sqrt{k}$ segments crossing the boundary of two square regions so that the two regions will not be merged. Therefore, we add at most $(2/3)(\sqrt{m} \times (\sqrt{m}/\sqrt{k}))$ data records

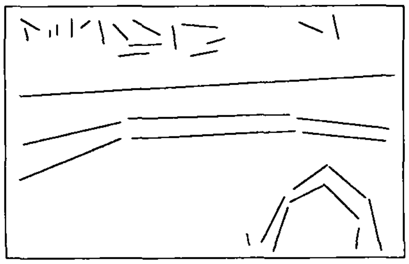
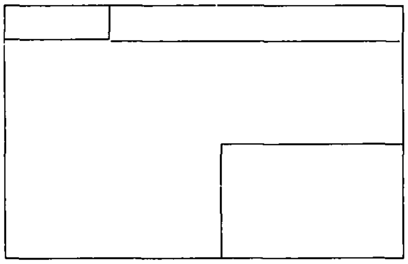


Image represented by linear segments



The image partitioned into regions for k=8

to the input data set. In the case of segments of arbitrary length r, the maximum number of all such segments in the image cannot exceed $(2/3)m_{max}$, else merging of regions will not take place. Thus, we need $(2/3)m/\sqrt{k}$ additional records in the worst case [48].

The partitioning strategy for stereo matching is similar to the above technique [48].

3.2 Parallel Implementations

The implementation of the image matching algorithm with respect to the partitioning strategy developed above, is based on the following idea: Assume we have a mesh connected computer with k processors and each processor has $O(\log m + k)$bit memory. During an iteration we examine a pair (i,j). A set of k objects belonging to a region and the labels of the corresponding region are fetched on the mesh. The $p(x,y)$'s are also fetched, so that the processor having a_z will also have $p(x,y_z)$, $1 \leq z \leq k$. For each label b the PE(0,0) constructs a record with the label b, the dimensions of the window $w(i,j,b)$ and a flag. Initially the flag is reset. PE(0,0) propagates this record to all the PEs in the mesh. Each PE containing a segment x examines the relation $(i,j)C(x,b)$ as well as $p(x,b)$. If $(i,j)C(x,b)$ is true and $p(x,b) = 1$ and the flag is reset it sets the flag. If the PE receives more than one record, it performs an OR operation between the two flags and considers the outcome of this OR operation as the new value of the flag. If the PE($k^{1/2}-1, k^{1/2}-1$) receives a record with the flag set it increases the value of $q(i,j)$ by 1.

The propagation of the record from PE(0,0) to PE($k^{1/2}$-$1, k^{1/2}-1$) is done such that PE(y,z) sends the record to the PE($y+1,z$) and to the PE($y,z+1$). The PEs along this line, which is parallel to the diagonal PE($k^{1/2}-1,0$)-PE($0,k^{1/2}-1$) read from their memory the $p^t(x,b)$'s. When a $w(i,j,b)$ extends further than the boundary of the region we add the record with b to a linklist pointed by the records of the adjacent regions (one linklist per region). At the end of each iteration the initial pointer of the linklist is updated to nil.

The record with the label b has its flag set to true when the PE($k^{1/2}-1, k^{1/2}-1$) receives the record with the $w(i,j,b)$ and its flag is set.

At the end of each iteration we examine q and we decide about the assignment (i,j).

Implementing the above strategy and pipelining the computations a speed up of $O(k)$ can be obtained. Details of the ideas presented here can be found in [48].

Our partitioning strategy also leads to a systolic implementation. The array consists of $k + \sqrt{k}$ cells (PEs). The PES are arranged in a 2-dimensional grid of size $(k^{1/2}+1) \times k^{1/2}$. Each PE has three ports A, B, C. A and B are the input lines and C is the output line. The B inputs of the PE($1,x$), $0 \leq x \leq k^{1/2}-1$, and the A inputs of the PE($y,1$), $1 \leq y \leq k^{1/2}$, are the inputs to the entire array. The PEs of the array use the C connection to propagate the results of the arithmetic/logic operations. The C inputs of the ($k^{1/2}$)th row are also used as control inputs. Each PE(p,q) receives, in its C input, the result of the operation which takes place in the PE($p+1,q$) $1 \leq p \leq k^{1/2}, 0 \leq q \leq k^{1/2}-1$. The difference is in the PEs of the first row (PE($0,z$) $0 \leq z \leq k^{1/2}-1$). There, each PE($0,z$) sends the result of its operation to the PE($0,z+1$). The PE($0,z$) receives in its B input the result of the C output of the PE($1,z$). The A inputs of a PE($0,w$) ($0 \leq w \leq k^{1/2}-2$) have a 0 as input. The A input of the PE($0,k^{1/2}-1$) is connected to the C output of the same PE.

The outline of an iteration is as follows: We fetch to the $k^{1/2}$ C inputs the object a_i and the label l_j. Then $k^{1/2}$ labels are fetched to the B inputs. We arrange these data items to be as in a transposed diagonal array of size $k^{1/2} \times k^{1/2}$. $k^{1/2}$ objects are fetched to the A inputs arranged in a diagonal array of size $k^{1/2} \times k^{1/2}$. This input sequence is repeated $\sqrt{k}$ times. Between two consecutive repetitions of this input sequence, we fetch $\sqrt{k}$ 0's to the A inputs. Following the labels array, there is an array of size $k^{1/2} \times k^{1/2}$ which contains the $p(x,y)$ relations, where a_x belongs to the set of the processed objects and l_y to the set of labels that are processed during this iteration. Between two consecutive rows of $p(i,j)$'s that we fetch to the A inputs, we fetch the label-set input. At the end of the iteration at the output of the PE($0,k^{1/2}-1$) we have $q(i,j)$.

The systolic array for $k^{1/2} = 3$ is shown in figure 3.2.

Speedup of $O(k^{1/2})$ can be achieved in the above systolic implementation. Details and proof of correctness of the above, as well as notes on the systolic implementation of the stereo matching can be found in [48].

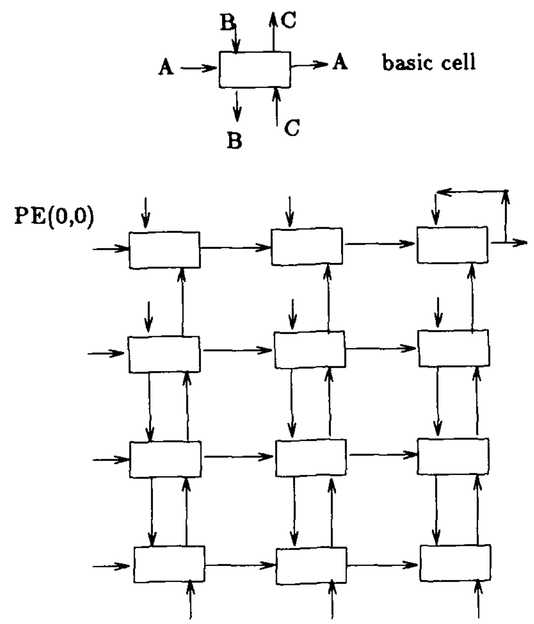


Figure 5: The systolic array for $k^{1/2} = 3$

References

[1] A. Agrawal, L. Nekludova, and W. Lim, *A parallel $O(\log N)$ algorithm for finding connected components in planar images*, Proceedings of the 1987 International Conference on Parallel Processing, 783-786.

[2] Hussein M. Alnuweiri and V. K. Prasanna-Kumar, *Optimal Image Computations on VLSI Architectures with Reduced Hardware.* Techn. Report USC, Institute for Robotics and Intelligent Systems, No 223, Sept. 1987.

[3] Hussein M. Alnuweiri and V. K. Prasanna-Kumar, *Optimal VLSI Sorting with Reduced Number of Processors.* Techn. Report USC, Institute for Robotics and Intelligent Systems, No 226, Dec. 1987.

[4] Hussein Alnuweiri and V. K. Prasanna Kumar *Efficient image computations on VLSI arrays with reduced hardware.* In proceedings of IEEE workshop on PAMI, 1987.

[5] M. Atallah and R. Kosaraju. *Graph problems on a mesh connected processor array.* JACM, vol. 3, pp. 649-667, 1984

[6] K. Batcher. *Sorting Networks and their Applications.* Spring Joint Computer Conference 32, pp 307-314, AFIPS PRESS, Montwole NJ, 1968.

[7] K. E. Batcher. *Design of A Massively Parallel Processor.* IEEE Trans. on Computers, C-29, Sep. 1980.

[8] S. H. Bokhari. *Finding Maximum on an Array Processor with a Global Bus.* IEEE Transactions on Computers, Vol. C-33, No. 2, February 1984, pp 133-139.

[9] D. Carlson. *The mesh with a Global mesh: A flexible high speed organization for parallel computation.* Tech. Report, Electrical and Computer Engineering Department, University of Massachusetts, 1985.

[10] D. M. Champion and J. Rothstein, *Immediate parallel solution of the longest common subsequence problem*, 1987 International Conference on Parallel Processing, pp. 70-77.

[11] F.Y. Chin, J. Lam and I.-N. Chen, Efficient parallel algorithms for some graph problems, *Communications of the ACM*, 25 (1982), 659-665.

[12] M. J. Duff. *A large scale integrated circuit image processor.* Proc. IJCPR, 1976.

[13] M. J. Duff. *Review of the CLIP Image Processing System.* National Computer Conf., Anaheim, CA, 1978.

[14] C. R. Dyer and A. Rosenfeld. *Parallel image processing by Memory Augmented Cellular Automata.* IEEE, Transactions on PAMI 1981.

[15] C. R. Dyer. *A VLSI pyramid machine for hierarchical parallel image processing.* Proc. IEEE conference on Pattern Recognition and Image Processing, 1981

[16] M. Furst, J. Saxe and M. Sipser. *Parity, Circuits and Polynomial Time Hierarchy.* Proc. IEEE Foundations on Computer Science, pp. 260-270, 1981.

[17] D. Gannon and L. Snyder. *Linear Recurrence Systems for VLSI: The Configurable, Highly Parallel Approach.* In Proceedings of ICPP, pp. 259-260, IEEE, 1981.

[18] R.L. Graham and F. F. Yao. *Finding the convex hull of a simple polygon.* Journal of algorithms 4, 1984.

[19] D. H. Greene and F. F. Yao. *Finite resolution computational geometry.* Proc. IEEE FOCS, 1986.

[20] S.E. Hambrusch and J. Simon, Solving undirected graph problems on VLSI, Tech. rep. CS-81-23, Computer Science, Penn. State Univ., 1981.

[21] D. S. Hirschberg, A. K. Chandra and D. V. Sarwate. *Computing connected components on parallel computers.* Communications of ACM, 1979.

[22] K. Hwang and F. A. Briggs. *Computer Architecture and Parallel Processing.* McGraw-Hill, 1984.

[23] J. Ja'Ja' and V. K. Prasanna Kumar. *Information Transfer in Distributed Computing with Applications to VLSI.* Journal of ACM, Jan. 1984.

[24] J. Ja'Ja', V. K. Prasanna Kumar and J. Simon. *Information transfer under different sets of protocols.* SIAM Journal on Computing, 1984.

[25] P. Kaufman, G. Medioni and R. Nevatia, *Visual inspection using Linear Features.* in Proc. Comp. Vision and Pattern Recognition Conf., Washington, DC, 1983.

[26] H. T. Kung and C. D. Thompson. *Sorting on a Mesh Connected Computer.* Comm ACM, 1977.

[27] F. T. Leighton. *Parallel computations using Mesh of Trees.* Technical Report, MIT, 1982.

[28] H. Li and M. Maresca. *Polymorphic-Torus Network.* Proc. International Conference on Parallel Processing, 1987.

[29] G. Medioni and R. Nevatia, *Matching Images Using Linear Features.* IEEE Trans. on Pattern Analysis and Machine Perception, PAMI-6, Nov. 1984.

[30] G. Medioni and R. Nevatia, *Segment-Based Stereo Matching* Computer Vision, Graphics, and Image Processing, 31, Feb. 1985.

[31] R. Miller and Q. F. Stout. *Convexity algorithms for pyramid computers.* Proc. 1984 International Conference on Parallel Processing.

[32] R. Miller and Q. F. Stout. *Computational Geometry on a Mesh-Connected Computer.* Proc. 1984 International Conference on Parallel Processing, pp. 66-74.

[33] R. Miller and Q. F. Stout. *Varying Diameter and Problem size in Mesh Connected Computers.* Proc. of the International Conference on Parallel Processing, 1985.

[34] R. Miller and Q. F. Stout. *Data Movement Techniques for the Pyramid Computer.* SIAM Journal on Computing, Vol. 16, No. 1, February 1987.

[35] R. Miller, V.K. Prasanna Kumar, D. Reisis, and Q.F. Stout, *Meshes with reconfigurable buses*, Proceedings of the MIT Conf. on Advanced Research in VLSI, 1988.

[36] D. Nassimi and S. Sahni. *Finding connected components and connected ones on a mesh connected parallel computer.* SIAM Journal on Computing, vol. 9, pp. 744-757, 1980.

[37] D. Nassimi and S. Sahni. *Data Broadcasting in SIMD Computers.* IEEE Transactions on Computers, C-30, Feb. 1981.

[38] D. Nath, F. N. Maheshwari, P. C. P. Bhatt. *Efficient VLSI networks for parallel processing based on orthogonal trees.* IEEE Transactions on Computers, 1983.

[39] R. Nevatia and K. R. Babu, *Linear Feature Extraction and Description* Comp. Graphics and Image Processing, vol. 13, 1980.

[40] R. Nevatia *Machine Perception* Prentice Hall, 1982.

[41] M. H. Overmars and J. Van Leeuwen. *Dynamically maintaining configurations in the plane.* Proceedings of the 12th Symposium on Theory of Computing, 1980.

[42] V. K. Prasanna Kumar. *Communication Complexity of various VLSI Models.* Ph. D. thesis, Dept. of Computer Science, Pennsylvania State Univ., 1983.

[43] V. K. Prasanna Kumar and C. S. Raghavendra. *Array Processor with Multiple Broadcasting.* Proceedings of the 1985 Annual Symposium on Computer Architecture, June 1985.

[44] V. K. Prasanna Kumar and M. Eshaghian. *Parallel Geometric algorithms for Digitized pictures on the Mesh of Trees organization.* International Conference on Parallel Processing, 1986.

[45] V. K. Prasanna-Kumar and Venkatesh Krishnan *Efficient Parallel Algorithms for Template Matching on Hypercube SIMD machines.* Proceed. of the International Conference on Parallel Processing, 1987.

[46] V. K. Prasanna-Kumar and Dionisios Reisis. *VLSI Arrays with Reaconfigurable Buses* Techn. Report USC, CRI-87-48 September 1987.

[47] V. K. Prasanna Kumar and D. Reisis *Parallel Image Processing on Enhanced Arrays* . International Conference on Parallel Processing, 1987.

[48] V. K. Prasanna-Kumar and Dionisios Reisis *Parallel processing of the Image and Stereo Matching* Technical Report USC-CRI, 1988.

[49] Dionisios Reisis and V. K. Prasanna-Kumar. *VLSI Arrays with Reconfigurable Buses* Proceedings of the International Conference on Supercomputing, June 1987, Athens, Greece.

[50] A. Rosenfeld and A. C. Kak. *Digital Picture Processing.* Academic Press, 1982.

[51] A. Rosenfeld. *Parallel Processors for Image Processing: 2-D arrays and extensions.* IEEE Computer, Jan. 1983.

[52] C. Savage and J. Ja'Ja, Fast, efficient parallel algorithms for some graph problems, *SIAM Journal on Computing*, 10, pp 682-691, 1981.

[53] M. I. Shamos, *Geometric complexity*. Proc. Annual Symp. on Theory of Computing, 1975.

[54] Y. Shiloach and U. Vishkin. *A $O(\log N)$ Parallel Connectivity Algorithm*. Journal of Algorithms, vol. 3, 1982.

[55] Q. F. Stout. *Mesh Connected Computers with Broadcasting*. IEEE Trans. on Computers C-32, pp. 826-830, 1983.

[56] Q. F. Stout. *Meshes with Multiple Buses*. Proc. 27th IEEE Symposium on the Foundations of Computer Science, pp. 264-273, 1986.

[57] S. L. Tanimoto. *A Pyramidal Approach to Parallel Processing*. Proc. of the International Symposium on Computer Architecture, 1983.

[58] C. D. Thompson. *Area Time Complexity for VLSI*. Proc. 11th Annual ACM Symposium on Theory of Computing, Atlanta, 1979, pp 81-88.

[59] P. S. Tseng, K. Hwang and V. K. PrasannaKumar. *A VLSI based multiprocessor for implementing parallel algorithms*. Proc. of the International Conference on Parallel Processing, 1985.

[60] L. Uhr. *Algorithm-Structured Computer Arrays and Networks*. Academic Press, 1984.

[61] L. G. Valiant. *Parallelism in comparison problems*. SIAM Journal on Computing, vol. 3, 1975.

[62] C.C. Weems, S.P. Levitan, A.R. Hanson, E.M. Riseman, J.G. Nash, D.B. Shu, *The image understanding architecture*, COINS Tech. Rept. 87-76, University of Massachusetts at Amherst.

[63] J. D. Ullman. *Computational Aspects of VLSI*. Computer Science Press, 1984.

[64] S. W. Wilson. *The PIXIE-5000 a Systolic Array Processor*. Workshop on Comp. Architecture and Image Database Management, 1985.

Parallel Hardware for Constraint Satisfaction

Michael J. Swain & Paul R. Cooper

Department of Computer Science
University of Rochester
Rochester, NY 14627

Abstract

A parallel implementation of constraint satisfaction by arc consistency is presented. The implementation is constructed of standard digital hardware elements, used in a very fine-grained, massively parallel style. As an example of how to specialize the design, a parallel implementation for solving graph isomorphism with arc consistency is also given.

Complexity analyses are given for both circuits. Running time for the algorithms turns out to be linear or $O(n^2)$, and if the I/O must be serial, it will dominate the computation time. Fine-grained parallelism trades off time complexity for space complexity, but the number of gates required is only $O(n^4)$.

1 Introduction

Constraint satisfaction is an important technique used in the solution of many artificial intelligence problems. Since the original applications such as Waltz filtering [Waltz 1975], an essential aspect of most constraint satisfaction algorithms has been their cooperative or parallel nature (eg. [Davis and Rosenfeld 1981]). While the parallel spreading activation nature of constraint propagation has been adopted whole-heartedly in specific applications such as connectionist relaxation [Feldman and Ballard 1982] [Hinton *et al.* 1984], some of the most complete and generally useful formal results analyze sequential algorithms [Mackworth and Freuder 1985]. Generating a formal analysis of one recent connectionist implementation of discrete relaxation [Cooper 1988] inspired us to design a massively parallel implementation of the classic, more generally applicable arc consistency constraint satisfaction algorithm, as described by Mackworth [1977] and Hummel and Zucker [1983]. The implementation is constructed of standard digital hardware elements, used in a very fine-grained, massively parallel style. The resulting circuit is thus an obvious candidate for fabrication in VLSI, and is thus similar to the work of Mead [1983].

The paper also provides a parallel hardware implementation of the arc consistency algorithm for a specific application - labelled graph matching. Such matching by constraint propagation and relaxation is often used in visual recognition systems [Cooper 1988] [Kitchen and Rosenfeld 1979].

Complexity analyses are given for both circuits. Running time for the algorithms turns out to be linear or $O(n^2)$, and if the I/O must be serial, it will dominate the computation time. Fine-grained parallelism trades off time complexity for space complexity, but the number of gates required is only $O(n^4)$.

2 Constraint Satisfaction

In this section, we review constraint satisfaction as formulated by Mackworth [1977] [1985] and Hummel and Zucker [1983]. A constraint satisfaction problem (CSP) is defined as follows: Given a set of n variables each with an associated domain and a set of constraining relations each involving a subset of the variables, find all possible n-tuples such that each n-tuple is an instantiation of the n variables satisfying the relations. We consider only those CSPs in which the domains are discrete, finite sets and the relations are unary and binary.

A k-consistency algorithm removes all inconsistencies involving all subsets of size k of the n variables. In particular, node and arc consistency algorithms detect and eliminate inconsistencies involving $k = 1$ and 2 variables, respectively.

More specifically, a typical arc consistency problem consists of a set of variables, a set of possible labels for the variables, a unary predicate, and a binary predicate with an associated constraint graph. For each i of the n variables, the unary predicate $P_i(x)$ defines the list of allowable labels x taken from the domain of the variables. For each pair of variables (i, j) in the constraint graph the binary predicate $Q_{ij}(x, y)$ defines the list of allowable label pairs (x, y). To compute the n-tuples which satisfy the overall problem requires that the local constraints are propagated among the variables and arcs.

Mackworth [1977] specifies one such constraint satisfaction algorithm for arc consistency: AC-3. In [Mackworth and Freuder 1985] it is shown that the complexity of AC-3 is $O(a^3e)$, where a is the number of labels (or the cardinality of the domain), and e is the number of edges in the constraint graph associated with $Q_{ij}(x,y)$.

Hummel and Zucker describe a parallel version of the arc consistency algorithm as follows (using Mackworth's notation).

Arc consistency is accomplished by means of the *label discarding rule:* discard a label x at a node i if there exists a neighbor j of i such that *every* label y currently assigned to j is incompatible with x at i, that is, $\neg Q_{ij}(x,y)$ for all $y \in D_j$. The label discarding rule is applied in parallel at each node, until limiting label sets are obtained.

3 A Hardware Implementation

The Arc Consistency (AC) chip consists of two arrays of JK flip-flops and suitable amounts of combinational circuitry. The most important part of the design is the representation for the two constraint tables $P_i(x)$ and $Q_{ij}(x,y)$. In the massively parallel connectionist design style, we adopt the unit/value principle, and assign one memory element to represent every possible value of $P_i(x)$ and $Q_{ij}(x,y)$. (As will be seen, JK flip-flops are used as the memory elements because of their convenient reset characteristics). For the hardware to be universal on any arc consistency problem, the two arrays must be able to represent any given $P_i(x)$ and $Q_{ij}(x,y)$ of sizes bounded by n and a.

The first (node) array consists of *an* flip-flops we call $u(i,x)$ which are initialized to $P_i(x)$. That is, if x is a valid label at node i, then the the flip-flop $u(i,x)$ is initialized to **on**. Thus initially at least, the flip-flops which are **on** all correspond to labellings of a node which are valid considering only the local (unary) constraint at that node. Note that all flip-flops are initialized. The ultimate answer to the computation (which labels are arc consistent at each node) will be contained in this array at the end of the computation.

The second (arc) array consists of $a^2n(n-1)$ flip-flops we designate $v(i,j,x,y)$ which are initialized to conform to the arc constraint table $Q_{ij}(x,y)$. Note that the table $Q_{ij}(x,y)$ can designate three things. If $Q_{ij}(x,y) = 1$, then the arc (i,j) is present in the constraint graph and the label pair (x,y) is a valid labelling of the pair. If $Q_{ij}(x,y) = 0$, the arc (i,j) is again present in the constraint graph, but the label pair (i,j) in not allowed on that arc. But $Q_{ij}(x,y)$ might also just not be present in the arc constraint table, which indicates that there is no consistency constraint between nodes i and j. To account for the fact that $Q_{ij}(x,y)$ might be incomplete, $v(i,j,x,y)$ is initialized as follows:

if i is adjacent to j in the constraint graph

$$v(i,j,x,y) = Q_{ij}(x,y)$$

otherwise

$$v(i,j,x,y) = 1$$

Note that the arc array is static; it does not change throughout the computation.

The basic structure of the two arrays of flip-flops is shown in Figure 1.

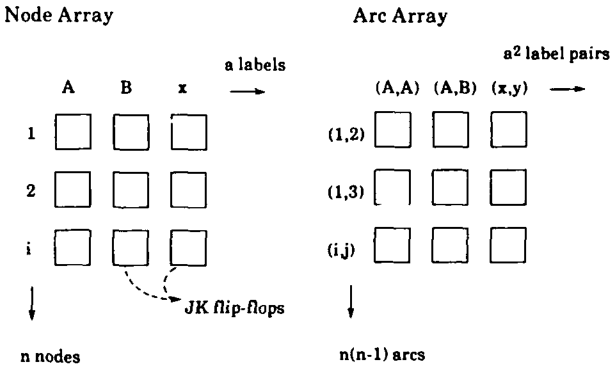


Figure 1: Unary and Binary Constraint Tables

It remains only to develop combinational circuitry which implements the label discarding rule - ie. that causes the flip-flop representing the label x at node i to be reset to zero if it becomes inconsistent. The combinational circuitry is thus designed so that the K (reset) input of the JK-flip-flop $u(i,x)$ receives the value:

$$reset(u(i,x)) = \neg \qquad {}_{j=1,j\neq i}\,{}_{y=1} (u(j,y) \wedge v(i,j,x,y))$$

The J input of each JK-flip-flop is tied to 0. A partial circuit diagram for this equation is given in Figure 2. This figure show the reset circuitry for one flip-flop in the node table $u(i,x)$. In the figure, the entire node table is present, but only the part of the arc table $v(i,j,x,y)$ useful for this node is drawn. An analagous circuit for each node completes the whole circuit.

To interpret the equation and circuit, consider first the inner term $u(j,y) \wedge v(i,j,x,y)$ for a particular case of $u(i,x)$. The fact that $v(i,j,x,y)$ is true tells us that there is an arc between i and j, and (x,y) is a consistent label pair for this arc. We already know that $u(i,x)$ is true; **anding** with $u(j,y)$ checks that the other end of the arc has a valid label. Point A on the circuit diagram in Figure 2 shows where this term is computed.

At this point, as far as node i is concerned, x is a label consistent with node neighbour j's label y. The $\bigvee_{y=1}^{a}$ simply ensures that at least *one* label y on neighbouring node j is .consistent. This function has been computed after the **or** gate at point B in Figure 2.

Label x on node i is thus consistent with its neighbour j. But what about node i's other neighbours? The $\bigwedge_{j=1,j\neq i}^{n}$ ensures that there is arc consistency among *all* node i's neighbour's. The **and** gate at C in Figure 2 ensures this.

If the signal is **on** at point C, that means that label x is consistent for node i - therefore, the flip-flop need **not** be reset. Thus the not gate.

To reverse the analysis, if some node j does **not** have a consistent labelling, then at point B, the signal will be **off**. The **and** will fail, so the signal at C will also be 0, and then the **not** gate will cause flip-flop $u(i,x)$ to be reset.

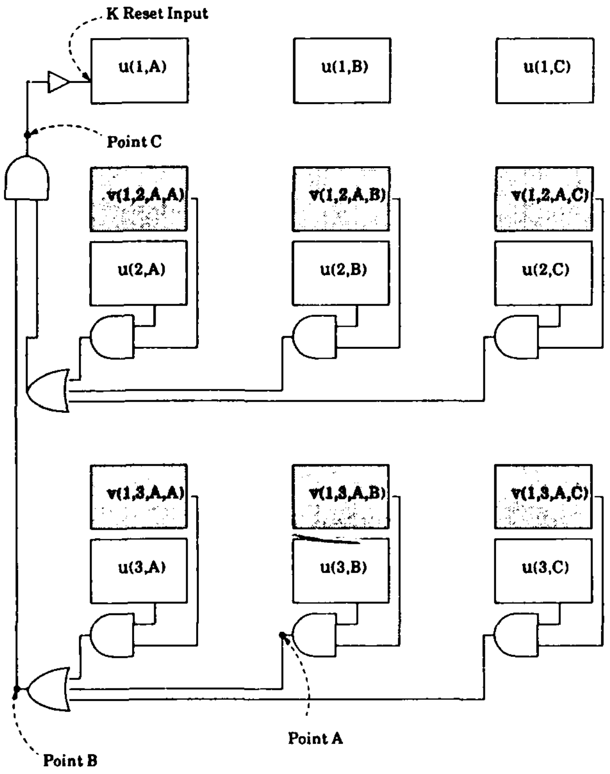


Figure 2: Partial Circuit Diagram for the AC Chip

3.1 Correctness

To begin with, recall that we are interested in discarding labels, an operation which corresponds to resetting **on** flip-flops to 0. Furthermore, since the J input of each JK-flip-flop in the node array is tied to zero, the flip-flops can only ever be reset to 0, never set. Once they are **off** they must stay off, so the whole process is clearly monotonic. Therefore, all we need to show for correctness is to show that the network correctly applies the label discarding rule. If the network discards labels when they should be discarded, and does not discard them when the should be kept, then it implements the label discarding rule correctly.

The label discarding rule can be formally expressed as follows:

$$\exists j(j \neq i)\forall y[u(j,y) \wedge v(i,j,x,y) = 0]$$

But this expression is equivalent to

$$\bigwedge_{j=1,j\neq i}^{n} \bigvee_{y=1}^{a} (u(j,y) \wedge v(i,j,x,y)) = 0$$

or

$$\neg \bigwedge_{j=1,j\neq i}^{n} \bigvee_{y=1}^{a} (u(j,y) \wedge v(i,j,x,y)) = 1$$

which is just the condition under which (i,x) is reset. Therefore, the network correctly discards labels when it should. The converse follows from negating the above equations.

3.2 Complexity

The circuit requires an JK-flip-flops for the node array, and $a^2n(n-1)$ flip-flops for the arc array. From Figure 2, we see that there is an **and** gate for every flip-flop in the arc array, so $a^2n(n-1)$ 2-input **and** gates are required for this purpose. For each of the an flip-flops in the *node* array there is $n-1$ **or** gates required, each taking a inputs - a total of $an(n-1)$ **or** gates. Finally, there are an **and** and **not** gates (**nand** gates), each taking $n-1$ inputs. There are also $O(a^2n^2)$ wires.

The worst case time complexity of the network occurs when only one JK-flip-flop is free to reset at a time. So if propagation through the **and** and **or** gates is considered instantaneous, the worst case time complexity is an. If a logarithmic time cost is assigned to the large fan-in **and** and **or** gates the worst case time complexity is $O(a\log(a)n\log(n))$.

Note that if the node and arc arrays must be initialized serially, loading them takes more time ($O(a^2n^2)$ steps) than executing the algorithm. For almost all applications of constraint satisfaction the binary predicate $Q_{ij}(x,y)$ can be specified with less than $O(a^2n^2)$ information, and so instead of the arc array a circuit could be built that supplies the correct values to the **and** gates without needing so many memory elements to fill. An application in which this is true is graph matching, which we describe in the next section.

4 Graph Matching

Graph matching can be defined as a constraint satisfaction problem. General graph matching requires k-consistency (and is NP-complete, in fact). With just arc consistency, a restricted yet still interesting class of graphs may be matched. Furthermore, the effectiveness of matching graphs by constraint satisfaction with only arc consistency can be enhanced if the graphs are labelled. This kind of restricted matching of labelled graphs is particularly suited to the visual indexing problem [Cooper 1988]. In this problem, labelled graphs are used to represent structurally composed objects. The constraint satisfaction process is used only to filter recognition candidates, and the few graphs not discriminable with the limited power of arc consistency can be addressed in other ways.

If labelled graph matching is framed as a constraint satisfaction process, the unary constraint is that the labels on corresponding vertices be the same. The binary (arc) constraint ensures that the connectedness between pairs of corresponding vertices be the same. In other words, if there is an edge between 2 vertcies in one graph, there better be an edge between the corresponding vertcies in the other graph. In this section, we describe without loss of generality the matching of undirected graphs.

So, for the graph matching problem:

$$P_i(x) = (\text{label}(i) = \text{label}(x))$$

and

$$Q_{ij}(x,y) = (\text{adjacent}(i,j) = \text{adjacent}(x,y))$$

For the graph matching problem the number of possible labels equals the number of vertices so $a = n$.

There are some modifications we can make to the general arc consistency circuit that are to our advantage for this particular application.

Constraint Table Computation by Special-Purpose Circuitry

One modification is to replace the arc array by a circuit designed as follows. Construct two arrays of

$$\binom{n}{2} = \frac{n(n-1)}{2}$$

flip-flops representing adjacent(i,j) and adjacent(x,y) respectively. Note that these are adjacencies in the input graphs, not in the constraint graph. For all possible $(i,j)(x,y)$ pairs, wire one flip-flop from the (i,j) array and one flip-flop from the (x,y) array to a gate computing the equality function $\overline{xor}$. Then the output of the $((i,j),(x,y))$'th gate represents $Q_{ij}(x,y)$. Then the network will have only $O(n^2)$ flip-flops to load prior to the computation.

Analogous special purpose circuitry to compute $P_i(x)$ from the vertex/label sets of each graph can easily be imagined as well. In the case, the equality gate must check equality of the labels, so is likely comparing more than just a single bit.

In any case, it is clear that actually computing the constraint tables $P_i(x)$ and $Q_{ij}(x,y)$ may be a significant part of the overall computation. In many specialized cases, it is clearly possible to actually build parallel circuitry to assist in computing the constraint tables, rather than serially computing the predicates beforehand and then loading them into the parallel hardware.

Symmetric Matching

Graph matching need not be simply isomorphism, as many vision applications emphasize [Shapiro and Haralick 1981]. If we restrict ourselves to pure isomorphism however, the graph matching problem is *symmetric*. In terms of the constraint satsifaction formulation, the symmetry means that the vertices of graph A have to be possible labels for graph B as well as vice versa. Therefore for a flip-flop (i,x) to stay, one may require it to be consistent regarding x as the label and i as the vertex and vice versa. So in addition to the **and-or** network described for the general constraint satisfaction problem the graph matching circuit has a complementary network in the opposite direction. The two circuits are **and**ed together before the inverter at the K input of the JK latch. Together these circuits compute

$$\neg\left(\left(\bigwedge_{j=1,j\neq i}^{n}\bigvee_{y=1}^{n}(v(j,y)\wedge Q_{ij}(x,y))\right)\wedge\left(\bigwedge_{y=1,y\neq x}^{n}\bigvee_{j=1}^{n}(v(j,y)\wedge Q_{ij}(x,y))\right)\right)$$

The circuit which implements this equation finds all possible labelings that are pairwise consistent both for matching graph A to graph B and for matching graph B to graph A.

4.1 Complexity

If no special purpose circuitry is used to compute$P_i(x)$ and it is input as a table of an or n^2 entries (in this case, $a = n$), then the complexity is as follows. The node array requires n^2 JK-flip-flops. The reduced arrays representing the input graphs require a total of $n(n-1)$ flip-flops. To replace the arc array, there are $n^2(n-1)^2$ $\overline{\textbf{xor}}$ gates. Analogous to the earlier design $n(n-1)$ 2-input **and** gates are required, $n^2(n-1)$ **or** gates, and n^2 **nand** gates. There are $O(n^4)$ wires, as for the general constraint satisfaction network.

The worst case time complexity for the graph matching network is the same as for the constraint satisfaction network, $O(n^2)$ ignoring propagation time and $O(n^2 \log^2 n)$ taking it into account. Loading and unloading the network takes $O(n^2)$ sequential time, and so does not affect the worst-case performance of the network. Since the expected time of the constraint satisfaction step could be much less than the worst-case performance, sequential loading and unloading is still likely to be the performance bottleneck.

4.2 Comparison with Connectionist Network

Cooper [1988] gives a connectionist network design for solving the same labelled graph matching problem addressed here. Interestingly, although the two networks were developed from completely different heritages, and for different reasons, they are remarkably alike. In particular, the central aspect of both designs - the representation of the unary and binary constraint predicates as completely filled-in tables - is exactlv the same. This reflects the adoption of the unit/value design principle, which is useful for obtaining a very high degree of parallelism, no matter what the primitive units of computation.

Unlike the digital network, the connectionist network is never intended to interface with sequential processes, and the input constraint tables are filled by parallel spreading activation. As a result, the I/O bottleneck does not occur. Of course, if the digital network receives parallel input, the same is true.

5 Discussion and Conclusions

The utility of constraint satisfaction methods in the solution of many AI problems suggests that efficient implementations might be widely useful. Furthermore, constraint satisfaction methods have an obvious parallel character.

In this paper, we have given a massively parallel design which provably implements one classic constraint satisfaction algorithm. Our implementation thus inherits the correctness characteristics of the original formulation. We have also shown how this design is easily specializable for particular problems. This specialization process provides a desirable alternative to designing and proving a new parallel network for each particular problem.

As might be expected, the highly parallel implementation runs very fast. Although technical worst case running time is linear in the number of variables, it is much more reasonable to expect that the network runs in a small constant number of time steps. Overall, if I/O time is not included, the performance of the network can be expected to be much better than that of the best sequential implementations.

It would be straight forward to construct our arc consistency chip, even for the general case. If, however, the parallel machine is forced to interface with sequential processes, the run-time complexity becomes similar to that expected from standard sequential implementations of arc consistency. The I/O bottleneck can be overcome by supplying parallel input or by specializing the chip to solve a particular problem, as we showed in the graph matching example.

Acknowledgements

This work was supported by a Canadian NSERC postgraduate scholarship, by the Air Force Systems Command, Rome Air Development Center, Griffis Air Force Base, New York 13441-15700 and the Air Force Office of Scientific Research, Bolling AFB, DC 20332 under Contract No. F30602-85-C-0008. The latter contract support the Northeast Artificial Intelligence Consortium (NAIC). We thank the Xerox Corporation University Grants Program for providing equipment used in the preparation of this paper.

References

[Cooper 1988] Paul R. Cooper, "Structure Recognition by Connectionist Relaxation", *Proceedings of the Image Understanding Workshop*, 1988.

[Davis and Rosenfeld 1981] L. S. Davis and A. Rosenfeld, "Cooperating Processes for Low-Level Vision: A Survey", *Artificial Intelligence*, 17:245–263, 1981.

[Feldman and Ballard 1982] J. A. Feldman and D. H. Ballard, "Connectionist Models and Their Properties", *Cognitive Science*, 6:205–254, 1982.

[Hinton *et al.* 1984] G. E. Hinton, T. J. Sejnowski and D. H. Ackley, "Boltzmann Machines: Constraint Satisfaction Networks That Learn", Technical Report CMU-CS-84-119, Department of Computer Science, Carnegie-Mellon University, 1984.

[Hummel and Zucker 1983] Robert A. Hummel and Steven W. Zucker, "On the Foundations of Relaxation Labeling Processes", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-5:267–287, 1983.

[Kitchen and Rosenfeld 1979] Les Kitchen and Azriel Rosenfeld, "Discrete Relaxation for Matching Relational Structures", *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-9:869–874, 1979.

[Mackworth 1977] Alan K. Mackworth, "Consistency in Networks of Relations", *Artificial Intelligence*, 8:99–118, 1977.

[Mackworth and Freuder 1985] Alan K. Mackworth and Eugene C. Freuder, "The Complexity of Some Polynomial Network Consistency Algorithms for Constraint Satisfaction Problems", *Artificial Intelligence*, 25:65–74, 1985.

[Mead 1983] Carver Mead, "VLSI and the Foundations of Computation", *Information Processing 83*, pages 271–274, 1983.

[Shapiro and Haralick 1981] Linda G. Shapiro and Robert M. Haralick, "Structural Descriptions and Inexact Matching", *IEEE-PAMI*, 3(5), 1981.

[Waltz 1975] D. Waltz, "Understanding Line Drawings of Scenes with Shadows", *The Psychology of Computer Vision*, pages 19–91, McGraw-Hill, 1975.

Scan Line Array Processors: Work in Progress

Allan L. Fisher , Peter T. Highnam and Todd E. Rockoff

**Carnegie Mellon University Department of Computer Science
Pittsburgh, Pennsylvania 15213**

Abstract

This paper describes the motivation for and progress toward the implementation and use of a special-purpose VLSI processor array for computer vision. A Scan Line Array Processor (SLAP) includes one processing element (PE) for each pixel on a single image scan line, arranged in a linear array. Each PE has internal fixed-point functional units and a register bank. A SLAP operates in SIMD fashion, with a system controller broadcasting a single instruction to all PEs in each cycle. Adjacent processing elements can exchange data in each cycle, and a dedicated video shift register performs concurrent image I/O.

The simple control structure and linear topology of a SLAP lend themselves to cheap, fast implementations and to easy scaling with technology improvements. Given this promise of great cost effectiveness, a key issue is the efficiency with which problems can be mapped onto the array. We have designed a number of algorithms and algorithm mapping schemes that demonstrate efficient mappings for a broad range of important image operations, both local and global.

In order to fully assess the practical value of the SLAP approach, we are constructing a prototype array with 512 processing elements. Built around custom 2 micron CMOS chips, the array and its controller will occupy three standard-size circuit cards and deliver some 4 billion 16 bit integer operations per second. Performance estimates on some commonly used algorithms indicate that this raw throughput can be put to good use. In addition to developing hardware and algorithms, we are also building low-level programming tools and an optimizing compiler for a high-level parallel language tailored for image processing. This paper gives an overview of the SLAP concept and the current implementation work, and provides some comparative performance predictions.

1. Introduction

Computer perception, particularly vision, demands extremely high computation rates from units that must be compact and inexpensive. Many parallel architectures for image processing and low-level vision have been constructed, and many more have been proposed. These architectures usually fall into one of three classes: those that attempt to operate on an entire image in parallel, like MPP (Batcher 1980, Potter 1985), CLIP4 (Duff 1976, Duff and Fountain 1986), DAP (Flanders et al 1977), and the Connection Machine (Hillis 1985), all of which operate as two-dimensional array processors; those that scan an image and operate on a small neighborhood window at each step, as do many industrial vision systems, the Cytocomputer (Lougheed and McCubbrey 1980), and PIPE (Kent et al 1985); and those with a structure not directly related to image structures or image computations, including systolic machines like Warp (Kung and Menzilcioglu 1984, Annaratone et al 1987), the NEC dataflow machine ImPP (Iwashita and Temma 1986), the Hughes HBA (Wallace and Howard 1987), and PASM (Meyer et al 1985). Elsewhere (Fisher and Highnam 1985) we have argued that the grid and small-window architectures have inherent drawbacks for image computations in terms of scaling for cost and performance. The more general purpose architectures, on the other hand, tend to exhibit a higher ratio of cost to performance than more specialized structures.

In this paper, we detail our initial explorations of an architecture that takes a position intermediate between the hypothetical full-image mesh and the neighborhood processors. The scan line array processor (SLAP) contains one processor for each pixel on a scan line, in the simplest case. This allows the architecture to scale gracefully with image size: the number of processors and the speed of each processor grow linearly with the perimeter of an image. We are aware of two other investigations along similar lines, the CLIP7 (Duff and Fountain 1986, Fountain 1985), which uses several rows of processors to emulate a full grid, and the

This research is supported by the Defense Advanced Research Projects Agency (DOD), ARPA Order No. 4976, monitored by the Air Force Avionics Laboratory Under Contract F33615-87-K-1499.

The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the US Government.

bit-serial PIXIE-5000 (Wilson 1985), a commercial product of Applied Intelligent Systems.

Using current VLSI technology, it is possible to build, at reasonable cost, a full scan line of reasonably powerful processors, avoiding the intricacies of bit-level programming and spatial image decomposition. Furthermore, the correspondence between the structure of the machine and the structure of an image allows most high-bandwidth traffic to take place independently of a general-purpose host or global bus, thus avoiding the need for expensive support hardware. Finally, as we demonstrate in this paper, SLAPs can efficiently execute a large set of important image computations and are good targets for current compiler technology.

2. Scan Line Array Processors

For current computer vision systems, typical image sizes are 512 by 512 eight bit pixels. In the future, image sizes are likely to increase with improvements in sensor technology and demands for higher precision in perception. However, the image processing grids of which we are aware are not capable even now of providing one processor per pixel. In fact, by sacrificing individual processor power (large grid machines have all been bit-serial) to permit the massive duplication of processor sites, the effectiveness of the programming model that these machines present for image processing has weakened. In this section we outline a processor structure that provides a good match with the problem, and motivate some of the architectural decisions described later.

When an image is moved between major components within a system, the transfer is of a stream of pixels. When the source of the image is a sensor then the stream comprises a sequence of rows. A SLAP operates on each row, or "scan line ' as it arrives, a distinct processor receiving each image column. The usual mode of operation for a SLAP is therefore to "sweep" down an image as shown in Figure 2-1. Each SLAP processor communicates with its two nearest neighors, forming a long linear array. Input and output of image rows is achieved by a video rate shift register built into the array. This allows a scan line to be shifted in at the same time as earlier scan lines are being processed and a result scan line is being shifted out, yielding online real-time processing.

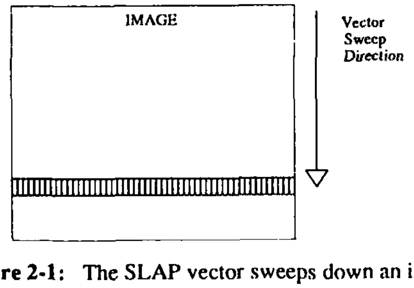


Figure 2-1: The SLAP vector sweeps down an image

The SLAP processor array uses the Single Instruction stream, Multiple Data stream (SIMD) control paradigm. In addition to minimizing overhead costs for instruction decoders, instruction memory, and control flow logic, SIMD control simplifies inter-PE communication because instruction execution is lock-step. This avoids the cost and delay of buffering and synchronization hardware. A SLAP's only high-bandwidth communication routes are the nearest-neighbor connections. Despite this simplicity of control and communication, we show below that a SLAP can be effectively programmed to perform a surprisingly wide range of tasks with high efficiency.

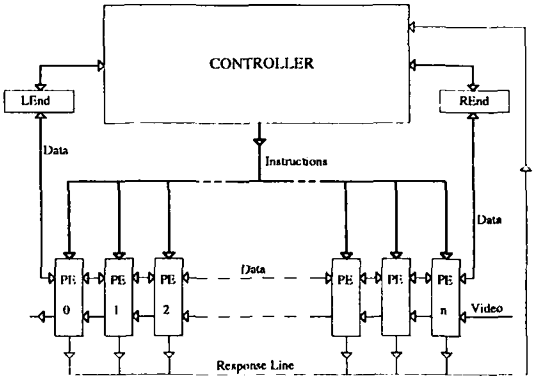


Figure 2-2: A SLAP system

Compared with a grid machine, a SLAP has only a moderate number of processors. This fact can be used to advantage to invest hardware resources in more powerful processing elements (PEs). Images rarely have binary pixels. Thus, the vast majority of arithmetic operations performed during the processing of an image are on multi-bit operands, generating multi-bit results. Therefore, the PE word size can, and should, be quite large. The disadvantage of a large word is that the PE datapath will not be fully utilized by some applications. On the other hand, programming a machine with a very small native word size requires the programmer to be aware of the maximum data sizes at each step during the computation, or simply assume and allocate space for the maximum size possible for each quantity from the beginning. The other extreme is to use a floating point representation. This we regard as being too expensive to implement in the context we have examined and unnecessary for most image computations. We see large integers as a good compromise between bit-serial and floating point processing for an image-processor.

The Achilles' heel of any SIMD machine is inflexibility. In order to provide for data-dependent computation, nearly every such architecture provides for conditional execution, whether centralized as in ILLIAC-IV (Hord 1982) or handled locally to each PE. Another important type of flexibility is provided by local address generation ability in each PE. This capability is found in ILLIAC and in other large-word SIMD machines. It is not found in bit-serial machines, for the simple reason that address generation, storage and communication would dwarf the rest of the processor. A third type of flexibility is the provision for single cycle shift and rotation operations with locally determined distances, crucial to flexible and efficient bit-field manipulations. All of these mechanisms are provided in the SLAP processor.

The controller of a SLAP system must be able to receive feedback from the vector for data reduction operations, associative operations or calculations with data-dependent runtimes. This is achieved in two simple ways: the controller provides two registers that the PEs at the vector extremes treat as neighbors, and any PE can write to a single bit "some/none" line that the controller reads. These mechanisms are cheap to implement and provide a channel for low-bandwidth results and a predicate evaluation ability for iterative operations. A controller is, of course, able to broadcast literals within the instruction stream.

This basic organization is very flexible. Cheaper, lower-performance machines may be constructed by using a shorter array and simulating many cells with one. For most algorithms, higher performance can be achieved by running several arrays in parallel or pipelined. Certain types of image operations can be facilitated by more frequent sampling of the video shift register. The machine can be used either online for real-time processing or offline. Real-time processing can be achieved even for some algorithms where the time to process a scan line is data-dependent, by asynchronously buffering several scan lines as necessary in the cells' registers.

The linear organization also offers great leverage in terms of scaling with technology or image size. Any number of processors can be packed onto a chip without changing its pinout, and the same applies to circuit boards. Systems of any reasonable size can be contructed by modular composition.

The array can be made more powerful by the addition of a variety of architectural features, at varying costs. Perhaps the most obvious is to supply each cell with enough memory to hold one or several image columns (as in CLIP7). This may be achieved (at current circuit densities, with only a few PEs per chip) with external memory, or with on-chip memory. If external memory is used, a cost/utility tradeoff between broadcast and individual memory addresses exists.

Depending on the application, a SLAP can be used in a variety of system configurations. The simplest is as a dedicated image processor/enhancer, where the array is controlled by a code ROM and sequencer, and all I/O is digital video. Another step up in complexity is an image processing system based on a host computer. Here the SLAP is provided with code in a RAM that can be written by the host, and video I/O is performed by frame buffers that are also accessible to the host. A fully configured vision subsystem might add a more complicated microcoded array controller and interfaces to the array's data ports. A key cost feature of all of these configurations is that the video data stream, the only necessarily high-bandwidth path in the system, is implemented with point-to-point links and regular addressing patterns. The host bus and host memory, usually the bottlenecks in a general-purpose system, receive only low-bandwidth filtered information from the array. Thus high-throughput vision processing can be performed on a very inexpensive host.

3. SLAP Algorithms

The success of a parallel architecture in serving a given application depends on how effectively the computations in question can be mapped onto the structure of the machine. The goal is to keep as many processors as possible busy doing useful work, thereby making cost-effective use of the hardware at hand. For a SLAP, the key issues bearing on its success are communication and control. The linear array structure has small global bandwidth (indeed, this is a major source of its economy of implementation), and hence algorithms involving completely general and unpredictable data access will not perform well. Also, a SLAP's SIMD control structure does not efficiently support highly heterogeneous computations. In this section we review results we have described in more detail elsewhere (Fisher 1986, Fisher and Highnam 1987), showing that a wide variety of vision tasks, including many that are usually thought of as being "global" in nature, can be performed efficiently on a SLAP.

3.1. Neighborhood Operations

Image to image transformations in which a pixel is replaced by a function of the pixels within a k by k window are easily implemented on a SLAP. Specific examples include convolution and median filters, together with the usual selection of neighborhood logic operations used in thinning and cellular automaton simulations. The row-by-row nature of the programming model requires that values and partial results be cached over several rows within each PE. Table lookup transformations can be achieved within a SLAP that has very limited memory at each PE by spreading disjoint segments of the table across a number of adjacent PEs or repeated broadcast of the complete table.

3.2. Region Operations

Many vision tasks, particularly in industrial vision, depend on finding, labeling, and extracting parameters of image regions. Regions may defined by intensity, local gradients, range data, or texture, for example. Because there is only a limited amount of time (in general) available for the processing of each image row, those tasks that require substantial lateral transmission of data are not easily handled by a single straightforward sweep as described earlier. Instead, a variety of techniques are possible, including: post-processing of a limited amount of information on the host; processing the image in vertical strips, the demarcation lines corresponding to maximal lateral transmission distances; sweeping the image transposed; and receiving a column-staggered version of the image (two pixels adjacent in the same image row, are presented to the vector in successive rows, columns unchanged). The common theme is that the vector and its controller perform the bulk of the computation with the assistance of a flexible memory controller, leaving high-level processing for the host.

3.3. Projections

Many important operations can be viewed in terms of projections along various curves or lines (Sanz and Dinstein 1987). Of particular interest, however, are linear projections; the linear Hough transform (Duda and Hart 1973), widely used for finding lines in images, consists in essence of projections along sets of lines oriented at angles from zero to π. On a SLAP, linear projections are easily computed by shifting accumulators from neighbor to neighbor along the lines of projection as the image passes through the array. For angles between $\pi/4$ and $3\pi/4$, each accumulator moves no more than one pixel horizontally for each scan line. For shallower angles, extra cycles can be devoted to the necessary shifting or the transpose of the image can be processed in a second pass. The SLAP prototype under construction can perform a Hough transform with ninety angular values in a single real-time pass (Fisher and Highnam 1987). Other projections, including the generation of a two-dimensional view of a three-dimensional scene, have been cast in terms of one-dimensional operations (Robertson 1987) and can be efficiently performed on a SLAP.

3.4. Image Transforms

A number of image processing applications call for 2D image transforms, such as the Fourier, Walsh-Hadamard and Haar. These are all separable, facilitating their efficient implementation on a SLAP endowed with enough memory per PE to accomplish a one column fast transform internally. Such a SLAP can perform an arithmetically simple transform within two frame times, with the time taken dominated by the time to tranpose intermediate results. The time taken by transforms requiring a great deal of arithmetic processing per output point is dominated by arithmetic, leading to very high processor utilization.

3.5. Miscellaneous Algorithms

- Skeletonization uses a local neighborhood operation iteratively to reduce "black" areas of an image to their centerlines. More efficient on a SLAP is a grassfire (Montanari 1968) algorithm, using two passes over the image to compute the minimal distances of each black pixel to a white one, then a pass thresholding to preserve local maxima. A SLAP implementation would use column-staggering or transposition.
- Histogramming is a straightforward operation on a SLAP. The accumulation is shared between PEs during the image pass; a post-pass phase accumulates histogram bins and shifts the histogram out of the vector.
- Image resampling on a SLAP is straightforward provided selective line fetch (under the command of the controller) or large column memories are available.
- Image warping has an efficient SLAP implementation when independently addressable column memories are available. Vertical displacements are handled by addressing, while horizontal displacements are handled by shifting.
- Operations that involve multiple images in pixel-to-pixel operations can be achieved by interleaving image rows (for example) as they are delivered.

3.6. General Comments

In addition to these general algorithm classes, many other problems seem to be well-suited to SLAP computation. In general, a SLAP is more flexible than strictly local image processors in that state related to contiguous features of an image can be maintained and passed among nearby processors as necessary. Global transfer of information can be achieved via the fast shift register and the broadcast SIMD control, allowing flexibility far beyond that of most window processors or nearest-neighbor structures. The addition of sufficient memory to hold an image column at each processor opens the door to a class of algorithms that neither window processors nor two-dimensional arrays can use effectively. Some general approaches to algorithm design for a linear array are discussed in an earlier paper (Fisher 1986).

4. SLAP Programming and Compilation

This section discusses some of the issues involved in programming a SLAP. We begin with a brief review of SIMD programming languages. Following a description of the loci of program control in a complete system, we discuss the SLAP programming environment that we are constructing. An interesting feature is a formulation of communication in fine grain systems that facilitates a high degree of automatic code optimization.

4.1. SIMD Programming

In terms of hardware investment, the SIMD paradigm provides a simple way to deliver a large number of processing cycles. However, realizing that potential without laborious hand-coding is problematic. General purpose machines support the common high level languages reasonably efficiently. They can do this because the abstract machines that underlie those languages are not too dissimilar, and those same abstractions are mapped easily to conventional processors. Another way of saying this is that the programming models presented by each high level language are easily mapped to conventional processors.

The programmers of SIMD machines do not have high level, multi-machine languages to choose from. This is because the machines constructed to date have been sufficiently diverse in both structure and intended application that an efficient mapping from even one, fixed, language to several machines, has not been seen.

Generally, SIMD systems have followed the same route in language definition by augmenting existing high-level languages with constructs that make it quite clear to the compiler which code should be executed in the PEs and which in the controller/host. These include the various Fortran and Algol implementations on the Illiac IV (Hord 1982), the Pascal implementation on the MPP (Reeves 1984), Connection Machine Lisp and C (Christman 1983, Steele and Hillis 1986), C on the PIXIE-5000 (Wilson 1985), Fortran on the DAP (Flanders et al 1977), C on CLIP4 (Duff and Fountain 1986), and C on PASM (Kuehn and Siegel 1985). Unfortunately, as revealed by user surveys (Perrott and Stevenson 1981, Wetherell 1980), and user reports (Marks 1980, Oldfield and Reddaway 1985, Smith 1984), a user must still be aware of the dimensions of the machine and the data.

4.2. SLAP Code Generation

Programming a SLAP involves generating code for the PEs, the controller, and for the host. The synchronous nature of a SLAP's communications facilitates the compiler generation of interdependent threads of computation in the vector and its controller from a single program. The host's program interacts with the vector and its controller only occasionally and asynchronously with the pixel data stream, so a subroutine or coroutine protocol is appropriate.

The SLAP component that receives host commands is the controller. The controller contains the code for the PEs and itself in separate memory units. When an image operation is invoked, the controller begins executing and issuing code from a certain address. The controller has branching and computational abilities of its own, which are employed concurrently with PE instruction execution to execute the usual constructs of a sequential language: conditional branching, iterative loops, and so on. The controller interacts with the vector using data as well as control, via the two "end" registers. Global vector status can be sampled using the some/none response line. Controller and PE cycles are synchronized, making the interchange of data easy. The controller can also transfer values that it has computed to all PEs by simply incorporating the value in the instruction stream as a literal. Note that there is no logical constraint preventing the controller and vector obeying different code for each input line.

The PEs are pipelined, but are not so complex that the PE instruction stream width is too small to yield precise control of the activities of PE components. This, with the symmetry of the functional units, makes PE code generation relatively straightforward. The impact of a small internal memory, local addressing, and the conditional capabilities have yet to be fully explored. The word size is expected to suffice for most operands, but multi-word operands are supported.

4.3. SLAP Programming

Following in the model of previous SIMD systems, we are implementing a language that is an extension of sequential languages, in our case these are Pascal, and C. The division of labor between the array and the controller is implicitly specified by the use of variables declared to be instantiated in the controller or the PEs. The compiler will generate code for the entire system. Within this framework two programming models are supported. The first is a "native" mode for the SLAP, code is specified on a scanline by scanline basis. The alternative is to write the program in a position-independent style. A large proportion of image computations can be expressed as implicitly parallel position-independent operations, accessing values at other pixel positions using relative offsets (Hamey et al 1987). For obvious reaons, this model is a good match with a SIMD machine, especially one where PEs are directly correlated with image coordinates. On a SLAP, a program written in the position-independent style is compiled into the code executed by the vector and the controller during each row.

To facilitate the coding of position-independent programs and establish a basis for efficient code optimization, we introduce a new type of unary operator to a conventional expression language, the *directional* (Fisher and Highnam 1988). In image processing we use four directionals **LEFT**, **RIGHT**, **UP** and **DOWN**. The application of a directional such as **LEFT**, to an expression *e*, means that the computation at the current position uses the value of *e*, as computed at the pixel position to the immediate left. Similarly for the other three directionals. Thus, for example, **LEFT UP** *e*, yields the value of *e* as computed at the upper left neighbor. These semantics can be formalized by defining the usual arithmetic operators on grids of values, and defining directionals as coordinate translations. The formulation of an algebra of communication and computation is then straightforward.

A drawback to a simple-minded implementation of position-independent code is the inability to reuse intermediate results from other pixels. For example, convolution with a symmetric kernel or sorting within a neighborhood involve computing values that can be useful at a number of distinct pixels. The directional formulation allows an expression graph optimizer to detect redundant computation and, using the algebra of operators, reorganize the graph to produce efficient code.

The optimizing component of a compiler for a grid *or* a SLAP architecture has been constructed using OPS5 (Forgy 1981, Brownston et al 1985) to manipulate data-dependency graphs. The input graph is derived in the usual way from an expression that includes all the usual arithmetic operators and directionals. The graph is transformed to minimize the total number of operations contained (including directionals). The compiler algorithm is greedy (in the technical sense) and makes heavy use of associative properties of operators. Results have been extremely promising. For example, the code for a symmetric convolution:

```
global  a,b,c;
result :=   a * LEFT UP p    +
            b * UP p         +
            a * RIGHT UP p   +
            b * LEFT p       +
            c * p            +
            b * RIGHT p      +
            a * LEFT DOWN p  +
            b * DOWN p       +
            a * RIGHT DOWN p ;
```

is transformed to:

```
temp1   := p * a ;
temp2   := p * b ;
temp3   := temp2 + UP temp1 + DOWN temp1 ;
result  := RIGHT temp3 + LEFT temp3 +
           DOWN temp2 + UP temp2 + p * c ;
```

The optimizer reduces a program containing 12 communications, 8 additions and 9 multiplications into one using 6 shifts, 6 additions and 3 multiplications. On a SLAP doing single-precision accumulation and using 8 bit pixels, this amounts to reducing about 44 instructions to about 18. The savings grow as precision increases, since multiplication is a multi-cycle operation.

The Canny edge operator has the property that the convolutions used in its computation are separable. Coding this operator, with a minimal window size of nine pixels, and then running the result through the optimiser produced very good results: multiplications reduced from 26 to 8; additive operations down from 17 to 9; communication steps reduced from 24 to 8. The optimizer produces even better results on larger windows.

Code using directionals is appropriate for direct execution on a grid. For a SLAP, the **UP** and **DOWN** directionals can be interpreted as temporal rather than spatial operators to yield a space-time processing schedule. Different operator interpretations can yield transposed, skewed and other execution schedules (Fisher and Highnam 1988).

5. Implementation

We are in the midst of constructing a prototype SLAP hardware and software system. The centerpiece of this effort is a custom CMOS chip including four processing elements and an instruction decoder. An array of these chips together with off-the-shelf memory will be driven by a global controller and interfaced to commercial video components and a host workstation. Finally, we are constructing a suite of program development tools including an optimizing high-level language compiler.

5.1. Array Chip

Each processing element, as shown in Figure 5-1, includes an eight bit video shift register stage and a sixteen bit datapath including a register file, an ALU, a rotate/shift unit and communication registers. Each PE also has a local control facility.

- The register file has 32 entries and is dual-ported, able to read and write a word in one cycle. The address for an access is presented either globally or from a local address register.
- The ALU performs a typical set of operations. A two-bit multiplication step and a one-bit divide step are provided, supported by a double-width accumulator set. Multiplication and division sequencing are supported in hardware, freeing the instruction stream for operations not involving the ALU. Multi-precision versions of the arithmetic operations are supported.
- The rotate/shift unit performs logical and arithmetic shift and rotate operations, with shift distances determined either locally or globally. A shift or rotate of any distance is completed in a single cycle.
- Adjacent PEs communicate via a communication register that can be transferred to (and loaded from) a neighbor in a single cycle. A PE can send a signal to the controller by writing a bit to the response line register, which is then be ORed with the response line register of every other PE.
- Conditional execution is achieved by causing processors to "sleep" or "wake" according to datapath condition codes. Sleeping processors are unable to change certain elements of their state. Support is provided for nesting of conditionals.

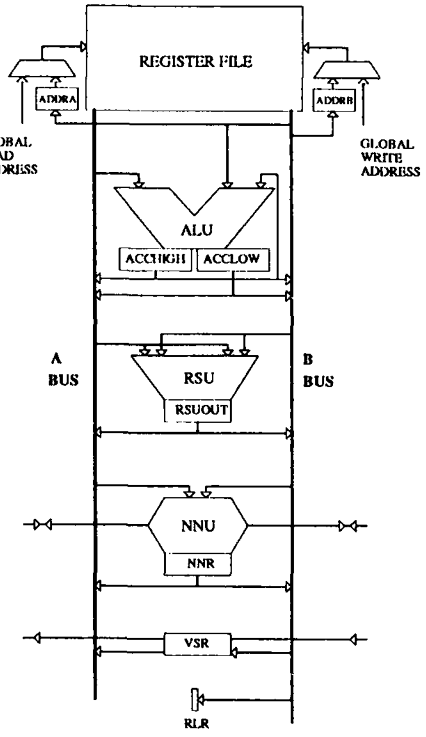


Figure 5-1: Processing Element data path

The array chip will comprise some 45,000 transistors, about 40,000 in the PEs and 5,000 in the decoder and peripheral circuits. In a 2-micron process, the chip will occupy a die area of about 63 square millimeters.

We expect the chip to achieve an instruction issue rate of 8 MHz, and a video shift register clock rate of 16 MHz. For a 512 by 512 frame scanned at 30 Hz, real-time processing on an array of 512 PEs allows some 500 instructions to be executed for each pixel. The 512-PE array has an aggregate computation throughput of 4 GOPS. For example, this allows a 9 by 9 convolution, an 11 by 11 pseudo-median filter, or a

21 by 21 edge finding operator to run in a single frame time of about 33 msec.

5.2. System Configuration

SLAP processor chips will be mounted on printed circuit boards along with commercial memory chips for column memories. Column memories are accessed using the video shift register for data and the neighbor communication path for addresses. Using one 32K by 8 static RAM chip for each array chip, we can supply each PE with 4K bytes, and allow one memory access for every four instructions. A separate board will hold the global controller and host interface. We expect to fit 64 array chips and their external memory onto a single standard-size Sun-3 VME card; thus a 512-processor system will require two array cards and a controller card. The system will also include commercial image capture, storage and display components.

5.3. Software

We have a simulator for a SLAP system that operates at the level of microcode with a time resolution of a half cycle. This software incorporates an assembler for the PE code. More work is needed on this program to provide increased logical error detection abilities, to permit the user to select the position on the simulation-emulation spectrum to use, to provide statistical information, and to refine the model of the vector controller and communication within and without the system.

Work is proceeding on the implementation of a higher level compiler that will generate code for both the vector controller and the processing elements. There will be two versions. The first will provide a more pleasant language than assembler for expressions and a few simple control constructs. The second compiler will incorporate the optimization ideas described above to generate efficient code without excessive contortions on the part of the programmer.

6. Performance

This section gives some performance estimates for the SLAP prototype under construction, and compares them to measured performance numbers for a VAX/11-780 and the CMU Warp machine (Annaratone et al 1987), a 10-processor floating point systolic array. VAX numbers were measured on tuned integer code, while Warp numbers were taken from the paper cited. The algorithms considered, all on 512x512 arrays, are:

- binary filtering: 3x3 neighborhood logical filtering of a binary image.
- histogram: a 256 level histogram.
- 3x3 convolve: 8 bit pixels and 8 bit weights. The SLAP code accumulates results to 16 bits precision.
- 3x3 median.
- Walsh transform: an FFT-like transform with weights of +/-1 on 8 bit pixels.
- Floyd-Warshall: all-points nearest neighbor in a graph. The SLAP implementation uses 32 bit integers. The Warp performance number has been scaled up from the 350 node figure given in the Warp paper.

Two caveats must be attached to these figures. One is that the Warp and VAX numbers represent measured performance on existing machines, while the SLAP numbers are best estimates based on handwritten code. The second is that the algorithms chosen are typically integer programs; thus Warp's floating point prowess is hidden under a bushel. Nonetheless, the numbers shown give a feel for SLAP's performance on some common image processing operations and place it in context.

Figure 6-1 shows elapsed times for each of the machines on each of the algorithms. Where the time indicated on SLAP is less than a frame time, it indicates the time needed to perform that operation either on a stored image or on an image already undergoing other processing. Note that the horizontal axis is logarithmic.

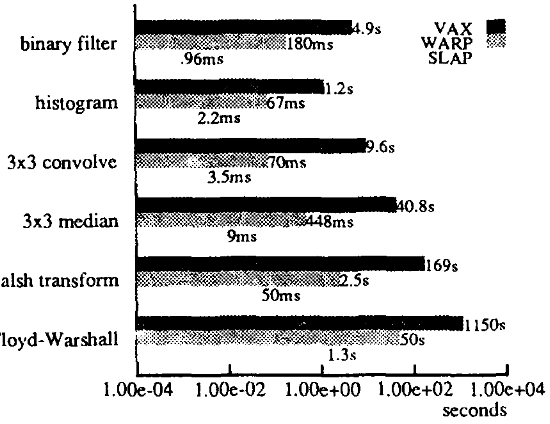


Figure 6-1: Elapsed times

Figure 6-2 places these numbers in context by normalizing them to a measure of raw power. Speedups relative to VAX range from 545 to 5104, with a geometric mean of 2166. Speedups relative to Warp range from 20 to 187.5 (though the second highest speedup is 50), with a geometric mean of 47.1 (35.7 with the high figure omitted). The vertical bar marked "OPS ratio" shows the speedup expected solely on the basis of peak operations performed per second. The scale factors shown rate SLAP at 4096 MOPS, Warp at 100 MOPS, and VAX at 1 MOPS. Where the speedup bars cross the line, SLAP performs even better than indicated by these ratios; for example, the SLAP datapath happens to be particularly efficient on the inner loops used for median filtering and binary filtering. Where the speedup bars lie to the left of the line, SLAP does not achieve its theoretical speedup. For histogramming, SLAP is able to achieve only a factor of 10 or so in useful concurrency, and hence does not fare as well as usual against a VAX. The lack of a parallel multiplier explains the subpar speedup for convolution. Finally, the Floyd-Warshall speedup shows the penalty incurred on SLAP by using 32 bit numbers, which on a large dataset require 4 accesses apiece to external memory.

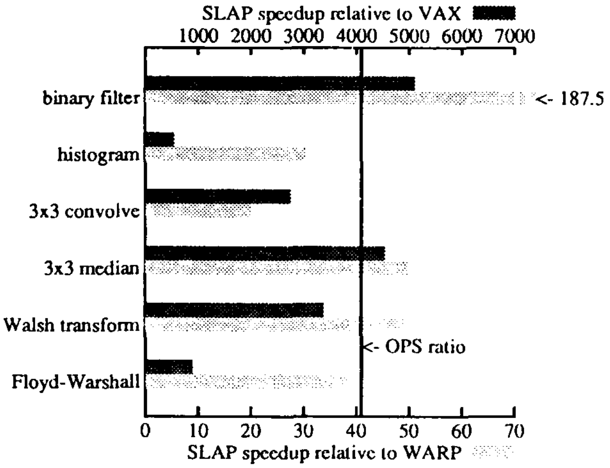


Figure 6-2: Scaled speedups

7. Summary

We have proposed a new architecture for low-level vision processing that is highly suitable for VLSI implementation, and have shown how it can be applied to a large set of problems. The SLAP architecture can be deployed in a variety of system environments, from dedicated real-time image enhancement boxes to a programmable vision subsystem for general-purpose computers. It can also be scaled over a wide range of cost and performance, according to user needs. As a result, SLAPs appear to offer high throughput in a large number of applications at modest cost.

Prototype hardware is under construction. Simultaneously, we are developing high and low level programming tools. Experimentation with realistic, complete image applications will provide a more complete evaluation of the SLAP concept.

8. Bibliography

Annaratone M, Arnould E, Gross T, Kung H T, Lam M, Menzilcioglu O and Webb J A 1987, *The Warp Computer: Architecture, Implementation, and Performance*, IEEE Transactions on Computers **C-36(12)**, December, 1523-1538.

Batcher K E 1980, *Design of a massively parallel processor*, IEEE Transactions on Computers **C-29(9)**, September, 836-840.

Brownston L, Farrell R, Kant E and N Martin 1985, *Programming Expert Systems in OPS5*, Addison-Wesley.

Christman D P 1983, *Programming the Connection Machine*, Masters thesis, Department of Electrical Engineering and Computer Science, MIT.

Duda R O and Hart P E 1973, *Pattern Classification and Scene Analysis*, Wiley and Sons.

Duff M J B 1976, *CLIP 4: A large scale integrated circuit array parallel processor*, 3rd International Joint Conference on Pattern Recognition, 728-733.

Duff M J B and Fountain T J 1986 (editors), *Cellular Logic Image Processing*, Academic Press.

Fisher A L and Highnam P T 1985, *Real-Time Image Processing on Scan Line Array Processors*, IEEE Computer Society Workshop on Computer Architectures for Pattern Analysis and Image Database Management, 484-489.

Fisher A L 1986, *Scan Line Array Processors for Image Computation*, 13th Annual International Symposium on Computer Architecture, 338-345.

Fisher A L and Highnam P T 1987, *Computing the Hough Transform on a Scan Line Array Processor*, IEEE Workshop on Computer Architectures for Pattern Analysis and Machine Intelligence, 83-87.

Fisher A L and Highnam P T 1988, *Communication and code optimization in SIMD programs*, Submitted for publication.

Fisher A L, Highnam P T and Rockoff T E 1987, *A VLSI SIMD Processing Element*, IEEE International Conference on Computer Design: VLSI in Computers and Processors.

Flanders P M, Hunt D J, Reddaway S F and Parkinson D 1977, *Efficient high speed computing with the Distributed Array Processor*, in High speed computer and algorithm organization, edited by D J Kuck, D H Lawrie and A H Sameh, Academic Press, 113-127.

Forgy C L 1981, *OPS5 User's Manual*, Technical Report CMU-CS-81-135, Computer Science Department, Carnegie Mellon University.

Fountain T J 1985, *Plans for the CLIP 7 Chip*, in Integrated Technology for Parallel Image Processing, Academic Press, 199-214.

Hamey L G C, Webb J A and Wu I C 1987, *Low-level vision on Warp and the Apply programming model*, in Parallel Computation and Computers for Artificial Intelligence, edited by J Kowalik, Kluwer Academic Publishers.

Hillis W D 1985, *The Connection Machine*, MIT Press, Cambridge, Massachussetts.

Hord R M 1982 (editor), *The Illiac IV, the First Supercomputer*, Computer Science Press.

Iwashita M and Temma T 1986, *Data Flow Chip ImPP and its System for Image Processing*, IEEE International Conference on Acoustics, Speech and Signal Processing, 785-788.

Kent E W, Shneier M O and Lumia R 1985, *PIPE (Pipelined Image-Processing Engine)*, Journal of Parallel and Distributed Computing, **2**, 50-78.

Kuehn J T and Siegel H J 1985, *Extensions to the C programming language for SIMD/MIMD parallelism*, International Conference on Parallel Processing, 232-235.

Kung H T and Menzilcioglu O 1984, *Warp: A Programmable Systolic Array Processor*, Real-Time Signal Processing VII, **495**, SPIE.

Lougheed R L and McCubbrey D L 1980, *The Cytocomputer: a practical pipelined image processor*, 7th International Symposium on Computer Architecture, 271-277.

Marks P 1980, *Low level vision using an array processor*, Computer Graphics and Image Processing, **14(3)**, November, 281-292.

Meyer D G, Siegel H J, Schwederski T, Davis N J IV and Kuehn J T 1985, *The PASM parallel system prototype*, IEEE Computer Society International Conference, Spring, 429-434.

Montanari U 1968, *A method for obtaining skeletons using a quasi-euclidean distance*, Journal of the ACM, **15(4)**, October, 600-624.

Oldfield D E and Reddaway S F 1985, *An image understanding performance study on the ICL Distributed Array Processor*, IEEE Computer Society Workshop on Computer Architectures for Pattern Analysis and Image Database Management, November, 256-264.

Perrott R H and Stevenson D K 1981, *Users' Experiences with the Illiac IV System and its Programming Languages*, ACM Sigplan Notices, **16**(7), July, 75-88.

Perrott R H, Lyttle R W and Dhillon P S 1987, *The Design and Implementation of a Pascal-Based Language for Array Processor Architectures*, Journal of Parallel and Distributed Computing, **4**, 266-287.

Potter J L 1985 (editor), *The Massively Parallel Processor*, MIT Press.

Reeves A P 1984, *Parallel Pascal: An extended Pascal for parallel computers*, Journal of Parallel and Distributed Computing, **1**(1), 64-80.

Robertson P K 1987, *Fast Perspective Views of Images Using One-Dimensional Operations*, IEEE Computer Graphics and Applications, **7(2)**, 47-56.

Sanz J L C and Dinstein I 1987, *Projection-Based Geometrical Feature Extraction for Computer Vision: Algorithms in Pipeline Architectures*, IEEE Transactions on Pattern Analysis and Machine Intelligence, **PAMI-9(1)**.

Smith K A 1984, *An image manipulation package for the DAP*, in Parallel Computing 83, edited by M Feilmeier, G Joubert and U Schendel, Elsevier Science Publishers BV (North-Holland).

Steele G L Jr and Hillis W D 1986, *Connection machine Lisp: Fine-grained parallel symbolic processing*, ACM Conference on Lisp and Functional Programming, 279-297.

Wallace R S and Howard M D 1987 *HBA: Built and Benchmarked*, IEEE Computer Society Workshop on Computer Architectures for Pattern Analysis and Machine Intelligence.

Wetherell C 1980, *Design Considerations for Array Processing Languages*, Software - Practice and Experience, **10**, 265-271.

Wilson S S 1985, *The PIXIE-5000 - A systolic array processor*, IEEE Computer Society Workshop on Computer Architectures for Pattern Analysis and Image Database Management, 477-483.

Pyramid Algorithms Implementation on the Connection Machine

Hussein A. H. Ibrahim
Columbia University,
Department of Computer Science
N.Y., N.Y. 10027

Abstract

Pyramid architectures have lately been proposed in the literature to implemen. a large number of image understanding algorithms, such as multi-resolution, pipelined, and bottom/up algorithms. In these architecture each processing element (FE) on an intermediate layer is connected to nine other PE's, four on the same level, four in layer below it, to a single parent in the layer above it. Hardware implementation of such pyramid machines is expensive because of the extensive wiring involved. The Connection Machine (CM), a highly parallel fine-grained SIMD machine with a hypercube and mesh interconnection networks, has been available for some time now to the vision community.

In this paper, we describe the implementation of an address mapping scheme that efficiently maps the pyramid architecture onto the Connection Machine (CM), resulting in an efficient way of implementing pyramid algorithms on the CM. In this mapping, each PE in the CM simulates a PE on the base of the pyramid, and at most one PE on the intermediate levels. Top/Bottom communication in the pyramid can be simulated in this scheme using only three hypercube communication cycles in the CM. On the other side, mesh communication occurring in all intermediate levels of the pyramid architecture can be simulated using at most a number of hypercube communication cycles that is proportional to the dimension of the cube (16 in the case of a 64K CM). A programming environment for pyramid algorithms, using this addressing scheme, has been implemented on the CM. It allows the user to create pyramid data structures, load/unload images from various pyramid levels, move data up/down, and perform several operations such as convolution and hierarchical operators on the created data structures.

1 Introduction

Image analysis tasks require a large amount of computation to process the large quantities of data contained in images with reasonable resolution. To meet this requirement, a number of parallel architectures [2], [3], [6], and [7], [5] have been proposed for application to image analysis problems.

Recently pyramid architectures have been proposed to implement efficiently (in real time) image analysis tasks, specially multi-resolution, and top-down/bottom-up image analysis tasks ([11], [3], [4], and [8]). Examples of algorithms that have been proposed for implementation on pyramid machines can be found in [1], [10], and [12]. The pyramid architecture consists of a set of mesh-connected layers of processing elements (PE's) successively decreasing in size by a factor of four.

Each PE on an intermediate layer is connected to four children on the layer below it, to a single parent in the layer above it, and to four (or nine) neighbors in the same layer, as shown in Figure 1. Hardware implementation of pyramid architectures with large number of PE's is expensive because of the large amount of wiring that is required in such machines.

The Connection Machine (CM), a highly parallel, fine-grained, single instruction stream, multiple data stream (SIMD) machine with mesh and hypercube interconnection networks, has been recently developed at Thinking machines Inc. and a 64K version has been available commercially.

In this paper, we describe an addressing scheme that maps efficiently the pyramid architecture onto the Connection Machine (CM). By efficiently here, we mean that pyramid communication can be simulated on the CM using a very small number of short fast CM cycles. This results in an efficient implementation of pyramid algorithms on the CM. In this mapping, each PE in the CM simulates a PE on the base of the pyramid, and at most one PE on the intermediate levels. Top/Bottom communication in the pyramid can be simulated in this scheme using only three hypercube communication cycles in the CM. On the other side, mesh communication occurring in all intermediate levels of the pyramid architecture

*This research has been supported in part by the Defense Advanced Research Projects Agency, under contract DACA76-86-C-0024

can be simulated using at most a number of hypercube communication cycles that is proportional to the dimension of the cube (16 in the case of a 64K CM). A programming environment for pyramid algorithms, using this addressing scheme, has been implemented on the CM. It allows the user to create pyramid data structures, load/unload images from various pyramid levels, display images on various levels, move data up/down, and perform several operations such as convolution and hierarchical operators on the created data structures.

In the following sections, we describe the addressing scheme, and show how various pyramid communication modes can be simulated on the CM. A brief description of the programming environment is then presented.

2 Mapping Scheme

The Connection Machine uses two modes of communication; the first one is mesh-communication where the whole array of PE's (64K) forms a two-dimensional orthogonal mesh (256 x 256). In this mode, each PE may communicate with its four neighbors in the east/west/north/south directions (NEWS network), as shown in Figure 2. This mode of communication is very fast as it uses direct links between the neighboring PE's.

The second mode of communication is the hypercube communication, in which each PE can communicate with all the PE's whose binary addresses differ from its own address exactly in one bit location. In the 64K machine, this means each PE is also connected to 16 PE's, using the binary 16-dimension hypercube interconnection network. For example, each PE communicates in the direction of the first dimension with the PE whose address is computed from its own address by flip-flopping the least significant bit (1 --> 0, 0 --> 1). In general, each PE communicates along the n-th dimension with the PE whose address differs from its own only in the n-th least significant digit. For example, PE0 is connected to PE1 along the first dimension, to PE2 along the second dimension, to PE4 along the third dimension, ..., and to PE(2**15) along the 16th dimension. Thus the maximum number of links that a message may require is equal to the number of binary digits in the PE's addresses (the number of dimensions in the hypercube). In the first version of the CM, when hypercube communication is invoked, the time allocated for the communication is the time needed for the message to travel from one node to its neighbor PE in the hypercube multiplied by the number of cube dimensions. This time must also take into account the buffering of messages that arrive at a specific node in the network at the same time. The second version of CM has a shorter hypercube cycle (petit cycle), that can be used when it is know that messages are not going to collide when traveling in the network. In our scheme, we define a new cycle time for hypercube communication. We will refer to this cycle as the "hypercube cycle". This is defined as the time required for communication between two neighboring nodes in the hypercube network. This cycle should be extremely fast (one nth of the petit cycle, where n is number of hypercube dimensions.)

To map the pyramid architecture on the connection machine, the CM mesh network of PE's simulates the base of the pyramid, while the PE's on the internal levels of the pyramid are going to be simulated according to the following scheme. The lower right PE of each 2 x 2 cube will simulate the parent of the four PE's in this 2-D cube. Similarly, for the second level above the leaf level in the pyramid, the PE connected to the PE in the lower right corner along the first dimension of the hypercube will be used to emulate the parent of the PE's in the 4 x 4 cube (4-D hypercube) A general formula that specifies the CM addresses of the PE's on the intermediate levels of the pyramid is given as following:

> $4^{i}.n - 2^{i-1}$, where *i* is the level number,
> n changes from 1 to the number of PE's on that
> level ($2^{2.(8-i)}$).

Figure 3 shows this mapping for an 8 x 8 mesh. The hypercube connections are not shown in this configuration, but they exist according to the scheme described before.

The mapping scheme described above ensures that each PE in the pyramid intermediate levels is emulated exclusively by one PE in the CM (one to one mapping). The proof is as follows. The address mapping formula is $4^{i}n - 2^{i-1}$. On the same intermediate level, *i* is constant, thus changing *n* will result in a different address each time. Now to prove that no two intermediate level PE's are mapped to the same CM PE address, First assume that this is true. It means that for PE n1 on level i1, there exist a PE n2 on level i2 such that

$$4^{i1}n1 - 2^{i1-1} = 4^{i2}n2 - 2^{i2-1}.$$

The equation can be re-arranged as follows:

$$4^{i2}(4^{i1-i2}n1 - n2) = 2^{i2}(2^{i1-i2-1} - 2^{-1}).$$

or

$$2^{i2}(4^{i1-i2}n1 - n2) = (2^{i1-i2-1} - 0.5).$$

The left hand side of the equation is always an integer as i1, i2, n1, and n2 are all integers, while the right hand side is not. Thus the initial assumption is not true. This proves that this mapping is one to one.

Now there are two types of communication in the pyramid that need to be simulated, the top/down communication (parent/child communication), and the lateral communication in the intermediate levels of the pyramid.

3 Simulating Pyramid Intermediate Levels Mesh Communication

In this section, we describe how the mesh communication on the pyramid intermediate levels can be simulated on the CM using the addressing scheme introduced in the previous section. Note that Simulating the mesh communication on the base of the pyramid is performed directly using the NEWS network of the CM. In what follows we how a send-east operation within level one of the pyramid is simulated in the CM, and then generalize this procedure for other pyramid levels (communication in the other directions is performed similarly).

To simulate mesh communication within level one, the information is sent first forward along the third dimension. Sending information forward along the *n*th dimension means that only PE's whose *th* bit is 0, send their information to the PE's whose binary address is similar except that the *n*th bit is 1. Thus, sending forward along the third dimension will involve PE3 sending the information to PE7, but PE7 will not send its information to PE3. When PE7 sends its information to PE3, we will refer to that as sending the information backward along the third dimension. We will use the following notations to express these two types of hypercube communication.

-->3 (send information forward
along the third dimension.)

3<-- (send information backward
along the third dimension.)

Thus sending information forward along the third dimension will ensure that half the PE's on level one have sent their information to their east neighbors (PE3 to PE7, PE19 to PE23, ..., etc). Now for PE7 to send its information to PE19, it first sends the information along the 5th dimension to PE23, and then PE23 send it back along the third dimension to PE19. This results in half the remaining PE's sending their information to their east neighbors. In our own terminology this is represented as: (-->5 followed by 3<--). To continue sending the rest of information east, similar scheme is followed. The complete procedure to send east in the first level of the pyramid in a 16-dimension hypercube is given by the following figure:

-->3
-->5, 3<--
-->7, 5<--, 3<--
-->9, 7<--, 5<--, 3<--
-->11, 9<--, 7<--, 5<--, 3<--
-->13, 11<--,9<--, 7<--, 5<--, 3<--
-->15, 13<--,11<--,9<--, 7<--, 5<--, 3<--

Note that these communication steps can be pipelined in such a way that there will be no repetition of the same communication steps. To do that a temporary variable has to be created to prevent overwriting a traveling value. For example if we want to send the value of variable A to variable B. In any send forward operation the variable A is sent to variable B, while in a send backward operation variable B is sent to replace the variable B in the receiving PE. The pipelined version of the above procedure is as follows: During each of these communication steps all of the PE's are actively sending or receiving information. Note also that, the above scheme can be augmented easily to perform wraparound shifting as well. This is accomplished by a sequence of send backward communication instructions. For example, in the 16-dimension hypercube this starts with 15<--, where the A variable contents is sent to to Variable B backward along the 15th dimension, and then a sequence of

-->15	-->n-1
13<--, -->13	n-3<--, -->n-3
11<--, -->11	n-5<--, -->n-5
9<--, -->9	.. ..
7<--, -->7	.. ..
5<--, -->5	
3<--, -->3	2i+1<--, -->2i+1

n: dimension of the hypercube, i: level number.

send backward of the contents of variable B to variable B in the receiving PE (13<-- , 11<-- , ... , 3<--), except for the last one where the contents of variable B is sent to variable A only in the receiving PE's.

In general, simulating send-east communication within level i in the pyramid, involves following the above procedure except that we stop when information is sent forward along the the 2*i+1 dimension. That means that all the intermediate mesh communications in the pyramid can be performed simultaneously by disabling the receiving PE's on any level when they have completed receiving the required information. This can be achieved by storing in each PE the pyramid level number of the PE it simulates in the pyramid architecture.

The execution time of the procedure to simulate all pyramid mesh communications is then proportional to the number of dimensions in the hypercube.

Figure 4 shows the output from the implementation of this scheme for a 5-levels pyramid (16 x 16 CM). The implementation has been performed on the CMLisp simulator running on a Symbolic machine. In this simulation we assume no wraparound in the shift operation. The communication steps for east shift operation is shown for levels 1 through 4.

4 Simulating Top/Down Pyramid communication on the CM

To simulate the pyramid top/down communication on the CM, three hypercube communication cycles are required to simulate the level-to-level communication in the pyramid. For example, on the bottom-level of the pyramid, going one level up, each PE sends its message first forward along the first dimension, then forward along the second dimension. For example, PE3 is the parent of PE's 0,1,2,3, and PE0 sends its information to its parent node by sending it first to PE1, and then to PE3. In our notation that consists of two steps -->1, and -->2. Note that sending information from PE1 to its parent involves sending its information forward along the second dimension (--> 2), and sending the information from PE2 to its parent involves sending forward along the first dimension (--> 1) Similarly going from level 1 to level 2 involves communication forward along dimensions 3 and 4 respectively Thus, PE3 sends its information to PE7 and then P15. PE14 is the parent node of PE's 3,7,11,15. Thus, PE15 sends the information now to PE14 backward along the first dimension.

Remember that in the pyramid architecture, each parent is connected to four children on the level below it. We refer to these four children as UL(upper left), UR(pper right), LL(lower left), and LR(lower right). Thus, in general, to send information from an UL child on level i to its parent on level i+1, information is send forward first along dimension 2*i + 1, then forward along dimension 2*i + 2, and finally backward along dimension i. Note that going along dimension 0 means no operation. In our notations this is represented by the following sequence:

--> 2*i + 1
--> 2*i + 2
i <--

Sending the information from child UR to its parent is executed by the following sequence:

--> 2*i + 2
i <--,

while sending information from child LL to its parent involves the following sequence:

--> 2*i + 1
i <--

Sending information from the LR child to its parent involves only one communication step (i <--). It is clear that sending information from a single child to its parent executes in at most 3 steps.

Sending information from all UL children to their parents is executed by the following sequence:

--> 1 , --> 2 , 0 <--
--> 3 , --> 4 , 1 <--
--> 5 , --> 6 , 2 <--
...
--> n-1 , --> n , <-- (n-1)/2

Note that the first step in all sequences can be performed simultaneously in one hypercube cycle if each PE stores its level number and uses it to compute the address of the receiving PE. Thus, executing a global send parent kind of communication takes at most 3 cycles if each PE has its pyramid level number stored into it. Sending from all children to their parent executes in at most 8 cycles. In conclusion simulating the parent/child communication executes in fixed time regardless of the hypercube dimensions. The parent/children communication has been also implemented on the CMLisp simulator running on the Symbolics machine.

5 Pyramid Programming Environment

The addressing scheme described in this paper has been used in implementing a programming environment to implement pyramid algorithms. This environment allows the user to create pyramid data structures, load/unload images into various pyramid levels, move data up/down, display pyramid data structures on various levels, and perform several operations such as convolution and hierarchical operators on the created data structures. This programming environment has been written in *LISP, a parallel programming language that has been developed at Thinking machines Inc. to program the CM. The * prefix is used as a naming convention with functions to indicate that these functions are going to be executed by the CM hardware. Operators and data preceded by !! are parallel operators and data that are performed in all PE's of the CM. More details can be found in [9].

The following example shows how to use this environment to find edges using a simple multi-resolution thresholding scheme. Some of the functions used are *LISP functions, while those that perform pyramid operations are part of the programming environment.

```
(*defun *find-edges (pmd start-level
                     thresh dest-pmd)
; The following two functions defines
; pyramid data structures variables.
    (*defpmd 'parent-value)
    (*defpmd 'edge)

; The following function sets the pyramid
; variable on a specified level to
; a certain value
    (*pmd-set-level start-level
              'parent value  (!! thresh))

 ; "refine" is the thresholding function.
 ; It recursively goes down the pyramid
 ; from  start-level to level 0. At each
 ; level, if the parent's value is above
 ; the threshold, it calculates the edge
 ; value for this pixel, and store the
 ; value in the pyramid variable edge
 ; which is passed afterwards to its
 ; children as the parent value.

    (refine pmd start-level thresh)

; The function *pmd-set sets the value
; of the pyramid variable in all levels
; equal to a certain value.
    (*all (*pmd-set dest-pmd edge))

; The following functions deallocate
; pyramid variables
    (*deallocate-*defpmd parent-value)
    (*deallocate-*defpmd edge)
  ) ; end *find-edges
  (*defun refine (pmd level thresh)
    (*all (*when (and!! (level!! level)
                  (>=!! (*pmd 'parent-value
                                level)
                        (!! thresh)))
           (*pmd-set-level level 'edge
                  (edge-op!! pmd level))))
     (cond ( (/= level 0)

 ; send-level-children is an example of
 ; one of the communication primitives. It
 ; sends the value of the parent on
 ; a certain level to its children
        (*all send-level-children!! level
                  (*pmd 'edge level)
                  (*pmd 'parent-value
                             (- level 1))))
        (refine pmd (- level 1) thresh))
          (t 't))
  ) ;end refine

 ; the function edge operator uses the
 ; "shift-level!!"  function to compute
 ; the edge function value.
```

6 Conclusion and Future Work

In this paper, we have described a mapping scheme that efficiently maps pyramid communication on the CM. It has been shown that inter-level communication (top/down) is performed in 3 hypercube cycles. Also, mesh (intra-level) communication in all levels can be performed in a number of hypercube cycles that is proportional to the number of pyramid levels. A programming environment that is based on this scheme has been introduced briefly. Currently we are implementing many of the pyramid algorithms using this environment. These algorithms include parallel texture and stereo algorithms.

References

1. Antonisse, H. J. "Image Segmentation in Pyramids". *Computer Graphics and Image Processing 19* (1982), 367-383.

2. Duff, M. J. B. A Large Scale Integrated Circuit Array Parallel Processor. Proceedings of the IEE Conference on Pattern Recognition and Image Processing, 1976, pp. 728-733.

3. Dyer, C. R. A VLSI Pyramid Machine for Hierarchical Parallel Image Processing. Proceedings of the IEEE Conference on Pattern Recognition and Image Processing, 1981, pp. 381-386.

4. Dyer , C. R. Pyramid Algorithms and Machines. Multicomputers and Image Processing: Algorithms and Programs, Preston, K. Jr., and Uhr, L., eds., 1982.

5. Hillis, W. D. The Connection Machine. M. I. T. Artificial Intelligence Lab., September, 1981.

6. Kushner, T., Wu, A. U., and Rosenfeld, A. "Image Processing on ZMOB". *IEEE Transactions on Computers 31*, 10 (October 1982).

7. Potter, J. L. "Image Processing on the Massively Parallel Processor". *IEEE Computer Magazine 16*, 1 (January 1983).

8. Schaefer, D. H., Wilcox, G. C., and Harris, V.J. A Pyramid of MPP Processing Elements. George Mason University, 1984.

9. . *LISP release notes. Thinking machines Inc., June, 1987.

10. Tanimoto, S. L. Sorting, Histogramming, and Other Statistical Operations on a Pyramid Machine. University of Washington, August, 1982.

11. Tanimoto, S. L. A Pyramidal Approach to Parallel Processing. University of Washington, January, 1983.

12. Tanimoto, S. L. Algorithms for Median Filtering of Images on a Pyramid Machine. University of Washington, January, 1983.

Figure 1: The Pyramid Architecture

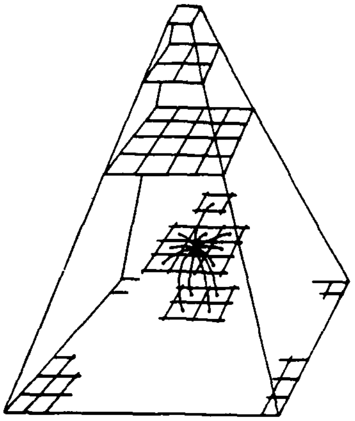
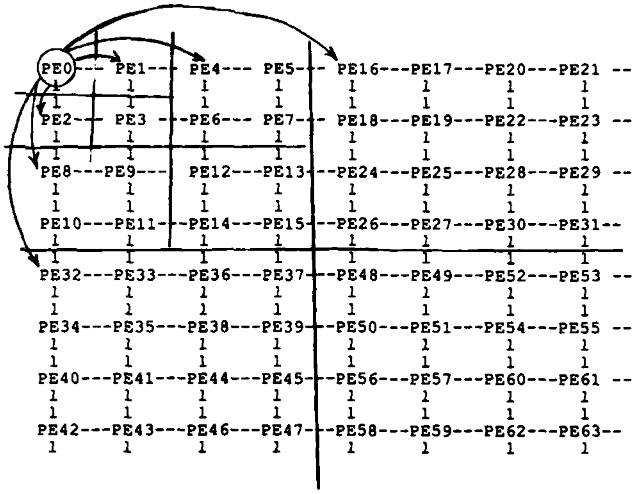


Figure 4: East Shift Operation

```
Random Values for non-leaf levels in 5-level pyramid
-------------------------------------------------------
 Level  1                               Level  2          Level 3  Lev.4
248 184 190 180 173 159 209  55          87 151 164 162   253  73  6
164 132  43  55  16  93 243  52          47 132 244 218   209  69
174 140 141 114  13  34 160 239         181 111 149 105
 74 156 177 205 171 114  37  82         212 248 150   9
222 185   7   0 240  35 161   9
203   3 110 140  43 130 204  44
 42 135   9  65  42 153 198  34
 23 171 242 185 159  20  15 173
                       Contents of the A variable

Shifting East Random Values for non-leaf levels ( A --> B)
------------------------------------------------------------
 Level  1                               Level 2          Level 3  Lev.4
Step 1:  --> 7    (A --> B)
  0   0   0   0 248 184 190 180   0   0  87 151   0 253     0
  0   0   0   0 164 132  43  55   0   0  47 132   0 209
  0   0   0   0 174 140 141 114   0   0 181 111
  0   0   0   0  74 156 177 205   0   0 212 248
  0   0   0   0 222 185   7   0
  0   0   0   0 203   3 110 140     PE's on levels 3,4 are
  0   0   0   0  42 135   9  65     disabled after this step
  0   0   0   0  23 171 242 185
                       Contents of the B variable

Step 2: <-- 5   (B --> B)
  0   0   0   0 190 180 190 180    0   0 151 151
  0   0   0   0  43  55  43  55    0   0 132 132
  0   0   0   0 141 114 141 114    0   0 111 111
  0   0   0   0 177 205 177 205    0   0 248 248
  0   0   0   0   7   0   7   0
  0   0   0   0 110 140 110 140
  0   0   0   0   9  65   9  65
  0   0   0   0 242 185 242 185
                       Contents of the B variable

Step 3: --> 5   (A --> B)
  0   0 248 184 190 180 173 159   0  87 151 164
  0   0 164 132  43  55  16  93   0  47 132 244
  0   0 174 140 141 114  13  34   0 181 111 149
  0   0  74 156 177 205 171 114   0 212 248 150
  0   0 222 185   7   0 240  35   PE's on level 2
  0   0 203   3 110 140  43 130   are disabled after
  0   0  42 135   9  65  42 153   this step.
  0   0  23 171 242 185 159  20
                       Contents of the B variable

Step 4: <-- 3     (B --> B)
  0   0 184 184 180 180 159 159
  0   0 132 132  55  55  93  93
  0   0 140 140 114 114  34  34
  0   0 156 156 205 205 114 114
  0   0 185 185   0   0  35  35
  0   0   3   3 140 140 130 130
  0   0 135 135  65  65 153 153
  0   0 171 171 185 185  20  20
                       Contents of the B variable

Step 5: --> 3     (A --> B)
  0 248 184 190 180 173 159 209
  0 164 132  43  55  16  93 243
  0 174 140 141 114  13  34 160
  0  74 156 177 205 171 114  37
  0 222 185   7   0 240  35 161
  0 203   3 110 140  43 130 204
  0  42 135   9  65  42 153 198
  0  23 171 242 185 159  20  15
                       Contents of the B variable
```

Figure 2: The address scheme of the Connection Machine PE's

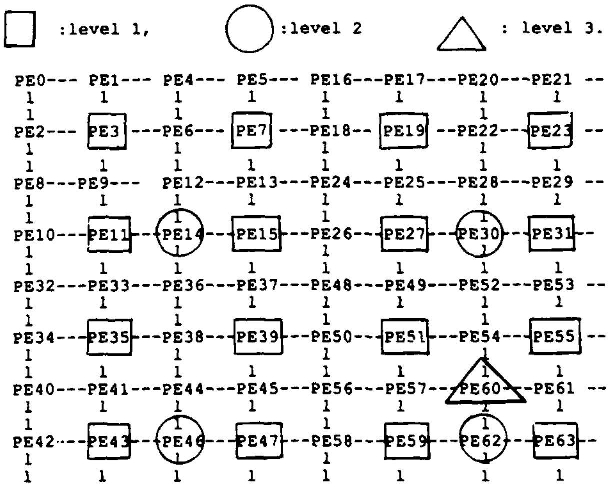

PE0--- PE1--- PE4--- PE5--- PE16---PE17---PE20---PE21 --
PE2--- PE3 ---PE6--- PE7--- PE18---PE19---PE22---PE23 --
PE8---PE9--- PE12---PE13---PE24---PE25---PE28---PE29 --
PE10---PE11---PE14---PE15---PE26---PE27---PE30---PE31--
PE32---PE33---PE36---PE37---PE48---PE49---PE52---PE53 --
PE34---PE35---PE38---PE39---PE50---PE51---PE54---PE55 --
PE40---PE41---PE44---PE45---PE56---PE57---PE60---PE61 --
PE42---PE43---PE46---PE47---PE58---PE59---PE62---PE63--

--- : Mesh Connections.

→ : An example of Hypercube Connections.

Figure 3: The layout of the pyramid intermediate levels

The original CM mesh represents level 0.


:level 1,
:level 2
: level 3.
PE0--- PE1--- PE4--- PE5--- PE16---PE17---PE20---PE21 --
PE2--- PE3 ---PE6--- PE7 ---PE18-- PE19 --PE22---PE23 --
PE8---PE9--- PE12---PE13---PE24---PE25---PE28---PE29 --
PE10---PE11---PE14---PE15---PE26---PE27---PE30---PE31--
PE32---PE33---PE36---PE37---PE48---PE49---PE52---PE53 --
PE34---PE35---PE38---PE39---PE50---PE51---PE54---PE55 --
PE40---PE41---PE44---PE45---PE56---PE57---PE60---PE61 --
PE42---PE43---PE46---PE47---PE58---PE59---PE62---PE63--

Robust Parallel Computation of 2D Model-Based Recognition

Todd Anthony Cass
Massachusetts Institute of Technology
Artificial Intelligence Laboratory

Abstract

An approach to 2D model-based object recognition is developed, suitable for implementation on a highly parallel SIMD computer. Object models and image data are represented as contour features. *Transformation sampling* is used to determine the optimal model feature to image feature transformation by sampling the space of possible transformations. It is shown that only a small part of this space need actually be sampled due to the constraints placed on possible transformations by individual matches of image features to model features. The procedure requires $O(Tmn)$ processors and $O(\lg^2(Tmn))$ time, where m is the number of model features, n is the number of image features, and T depends on the size of the image. The resulting procedure works well and is extremely robust in the presence of occlusion. An implementation of the procedure on the Connection Machine is described, and some experimental results given.

1 Introduction

We are interested in understanding how to perform model-based object recognition on a parallel computer. The primary motivation is that it may be possible to achieve extremely fast model-based recognition by exploiting parallel computation. The domain we consider in this paper is that of 2D object recognition from digitized images, based on object models. Figure 1 shows an example of the kind of problem we expect to solve. Here we have a typical input image of a scene including an instance of the modeled object and several unknown objects, and the output of the system indicating recognition of the object. The modeled object is shown in figure 9. This is the actual input and output of the recognition procedure as implemented on the Connection Machine parallel computer[13].

We have a well known problem with a new constraint: Develop a robust model-based recognition procedure that can be implemented efficiently on a parallel machine. The model-based recognition task is to find the pose of a known object in the image, which is equivalent to determining a transformation which aligns the model with its instance in the image. This is difficult for two major reasons. First, in a feature-based method, such as we develop here, it is difficult to determine the correct correspondences between model and image features since there are an exponential number of mappings from image features to model features. This is further complicated when some model features are absent from the image due to occlusion, and when there are extra image features due to the presence of unknown objects. Second, even given a correct matching of image features to model features, there is probably no unique transformation aligning the model with its instance in the image since there is uncertainty as to the exact location of image features due to sensing difficulties, and the fact that image features may be only fragments of their corresponding model features. These two problems conspire to make determining a valid transformation difficult. The technique that we shall develop here is based upon discretely sampling the space of possible transformations to determine the transformation best aligning the model with its instance in the image. As we will see, only a small fraction of the entire space need be sampled. This leads to a robust and fast parallel algorithm.

The difficulties of the recognition process itself are well known. A significant amount of work has been devoted to 2D model-based object recognition, and several good systems have been developed for serial computers[5] [8][6]. How might we do this in parallel, and why is developing a parallel procedure difficult? One approach is to convert an existing serial algorithm to a parallel one. However, it may not be possible to take a serial algorithm and modify it to work on a parallel architecture, taking full advantage of the available processing power. Recall that a problem of size M requiring $\Omega(f(M))$ time on a single processor machine, requires time $\Omega(f(M)/N)$ on a parallel machine with $O(N)$ processors. So at best we get a speedup by a factor of N. In practice, however, we often cannot achieve this speedup depending on the nature of the problem. As a point of interest, there are problems solvable in polynomial time on a single processor that, it is conjectured, cannot be solved in $O(\lg^k M)$ time for any k on a polynomial sized parallel machine [14]. So while a good parallel algorithm can be translated into a good serial algorithm, running a factor of $O(N)$ slower, by straightforward simulation of the parallel machine, it may not be possible to take a serial algorithm and achieve a speedup of N by parallel implementation.

As just mentioned, we seek the transformation aligning the model with the object in the image. We introduce the technique of *transformation sampling*. Simply stated, the 3D space of all possible transformations of the model is sampled on a 3D grid, where a point in this space represents a particular transformation consisting of a rotation, ϕ, about the origin and a translation, u and v, in position. This is a 3D space since we assume that scale is known. As we will see, it is not necessary to sample the entire space, but only regions corresponding to transformations which will keep some match of a model feature and an image feature near each other after transforming the model feature. Indeed, it is easy to see that there are many transformations which will result in none of the model features being near any image feature. These transformations can be safely ignored. This procedure is easily implemented on a parallel machine resulting in a fast, and quite robust 2D recognition procedure.

The transformation sampling technique bears resemblance to two existing recognition tools, clustering, such as the Hough transform[10][11][7], and global measure optimization. As we will show, transformation sampling does not have several of the characteristic shortcomings of Hough transform methods, and is efficiently implemented in parallel.

The contribution of this work is the development of the technique of transformation sampling, the description of a parallel algorithm for 2D model-based recognition, and the implementation

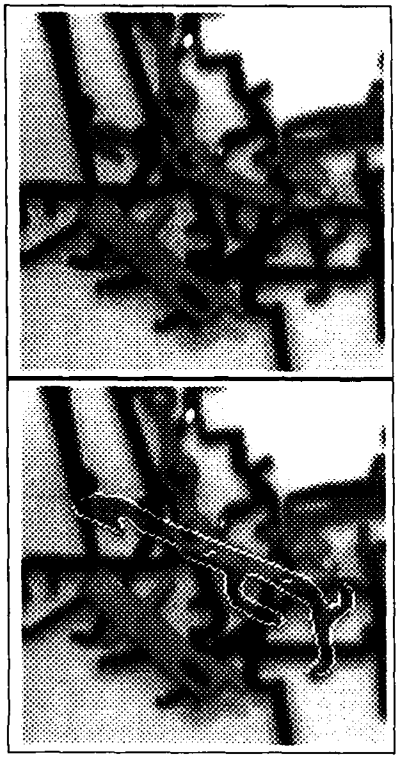


Figure 1: An example of an input image and the recognized object. The modeled object is shown in figure 9.

and demonstration of the procedure. Transformation sampling is key to the procedure, and is not subject to some of the problems with commonly used recognition techniques like the Hough transform. We will show that this procedure requires $O(Tmn)$ processors and $O(\lg^2(Tmn))$ time, where T depends on the size of the image, and m and n are the number of model and image features respectively. Given the assumption that the image size is constant, we can consider T constant.

We employed two very general criteria to guide development of this parallel recognition procedure: (1) The procedure must be robust in the presence of sensing errors, as well as the presence of several occluding objects. (2) The procedure should be implementable on a parallel computer so that it runs fast on a parallel machine, say $O(\lg^k M)$ time on $O(M^j)$ processors, where M characterizes the size of the problem. In short, it must work and run fast on a realizable machine.

2 Background

Before beginning the specific description of the recognition algorithm we need to briefly introduce some background material. This section provides some of the background concepts we will rely on as we develop the recognition algorithm. Here we describe the representation of models and image data; we formalize the notion of transformation of the model, and we discuss the concept of transform parameter space. This facilitates the formal definition of the recognition problem.

2.1 Models and Image Data

2.1.1 Features

The modeled objects, as well as the image data, are two-dimensional. Objects are represented by their boundary contours, which in turn are represented by a set of contour *features*. The contour features are characterized by a unique position and orientation. We will consider two types of features, *point features* and *extended features*. Point features simply consist of the orientation of the contour normal at a specific position, and are used primarily for the theoretical development of the algorithm. Extended features are simply straight line segments approximating sections of the contour, and are actually used in the experimental implementation. Both the stored model of an object and the objects in input image are represented by these features.

2.1.2 Uncertainty

We assume that noise in the image, spatial distortions in the imaging device, edge detection, as well as the polygonal approximation of the contour result in *uncertainty* in the position and orientation of each image feature. This means that the measured position and orientation of a feature in the image may differ from the actual position and orientation of the feature in the scene. We assume that this uncertainty is bounded, and let R denote the upper bound on the uncertainty of the position of an image feature, and Θ denote the upper bound on the orientation uncertainty of an image feature. Thus, considering a point feature for example, the true position in the scene of an image feature falls within a circle of radius R centered at the measured position, and the true orientation of the features is within Θ radians of the measured orientation. This follows closely the representation used by Grimson and Lozano-Pérez[8].

2.2 Transformations

Given a model of an object represented as a set of features, and a set of data features representing the input image, if the scene contains the modeled object there exists a transformation on the model which will align it with its instance in the image. If we let M_ϕ represent a 2D rotation matrix, $\vec{d}$ represent the position of an image feature, and $\vec{m}$ the position of its corresponding model feature, then $\vec{d} = M_\phi \vec{m} + \vec{t}$ for some ϕ and $\vec{t}$, where ϕ represents the rotation of the model feature and $\vec{t} = [u \quad v]^T$ is the translation in the x and y directions. Thus a transformation is parameterized by (ϕ, u, v) and consists of a rotation about some origin, followed by a translation.

When speaking of a transformation, we mean a *relative transformation* between a particular model feature and image feature. The term *feature match* will be used extensively to describe a particular pair of a model feature and an image feature.

Let $\{m_i\}$ describe the set of model features, and $\{d_j\}$ describe the set of image features. Then $\langle m_i, d_j \rangle$ describes a particular feature match and $S = \{\langle m_i, d_j \rangle\} = \{m_i\} \times \{d_j\}$ is the set of all possible feature matches. We say that a subset $S' \subseteq S$ is a *mapping* or a *matching* between model features and data features. Further, let $m = |\{m_i\}|$, and $n = |\{d_j\}|$.

2.3 Recognition

Since there is uncertainty as to the actual position and orientation of an image feature, when considering a particular feature match, any transformation which brings the model feature to within the uncertainty bounds on position and orientation of the image feature could be the correct transformation. We say that in this case the features *align* to within the uncertainty bounds. We say a set of feature matches, S', is *globally consistent* if there exists some transformation on the model features which simultaneously aligns all the matches in the S'.

Recognition consists of determining the presence of an object in the image, and of finding the pose of the object: position and orientation. Of course, the position and orientation of an object in the image are directly related to the transformation aligning the model with its instance in the image. In the next section we will associate a function $F(\phi, u, v)$ with each transformation. We will define recognition as finding a transformation producing an optimal value of $F(\phi, u, v)$. By appropriate definition of $F(\phi, u, v)$, recognition consists of finding large sets $S' \subseteq \{< m_i, d_j >\} = \{m_i\} \times \{d_j\}$ that are globally consistent, and the transformations at which the matches of S' align. Note that if an image feature and a model feature are incorrectly matched, but still align within the bounds for some transformation, then there is no way to distinguish them from correctly matched features, and they will be considered as such.

2.4 Transformation Parameter Space

For each match there is a range of rotations which will bring the orientation of the model feature to within the orientation uncertainty, Θ, of the image features orientation. For each of these rotations of the model feature, after performing it, there is a range of translations which will bring the position of the model feature to within the position uncertainty, R, of the image feature. Each transformation can be represented as a point (ϕ, u, v) in the the transformation parameter space $\mathbf{TP} = \Re^2 \times SO_2$ where SO_2 is the 2D rotation group, ϕ the rotation and u and v the translation in the x and y directions respectively. Thus, the set of valid transformations which leave the model feature aligned within the uncertainty bounds with the image feature form a region in parameter space we call a *match region.*

3 Recognition as Sampling Parameter Space

By appropriate choice of the function $F(\phi, u, v)$ characterizing transformations, recognition will consist of finding a transformation which aligns a large number of feature matches. This corresponds to finding the regions in the transform parameter space, **TP**, which intersect a large number of match regions when the match regions for all possible feature matches are constructed in **TP**. Such a region defines a mapping between model and image features, and provides a range of transformations which aligns each match in the mapping.

3.1 The Structure of Parameter Space

Consider each match $< m_i, d_j >$ as defining a scalar field in **TP**, which has value $f_{ij}(\phi, u, v) = c_{ij} > 0$ for (ϕ, u, v) within the match region and 0 outside it; where c_{ij} is a constant depending on the match $< m_i, d_j >$. Let the value of the field resulting from the superposition of the f_{ij} in **TP** be given by $F(\phi, u, v) = \sum_{i,j} f_{ij}(\phi, u, v)$. In this particular case $F(\phi, u, v)$ is a piecewise constant function which changes value only at the boundary of a match region. Call these piecewise constant regions *intersection volumes*. In general, we could define $f_{ij}(\phi, u, v)$ as any function quantifying properties of its associated match $< m_i, d_j >$; similarly, the value of $F(\phi, u, v)$ could be any function of the f_{ij}.

3.2 Finding the Optimal Transformation

Let an optimal transformation be defined as a transformation with a maximal value of $F(\phi, u, v)$. Finding the optimal transformation amounts to finding the intersection volume with an optimal value of $F(\phi, u, v)$. For convenience we will call the *value* of an intersection volume the value of $F(\phi, u, v)$ in the region. In fact, all we really need to do is find a sample point in parameter space which falls in this optimal intersection volume. Such a point defines the same mapping between model and image features as the entire containing intersection volume, and provides a globally consistent transformation.

3.3 Transformation Sampling

Part of the difficulty in determining a transformation lies in determining *which* model features and image features correspond. Some existing recognition systems approach the problem by first determining a set of corresponding feature matches, and then determining a transformation from these matches[8][5], but finding this matching is difficult. We accomplish both in one step, by finding a transformation that aligns a large fraction of the model with its instance in the image.

We have recast the recognition problem as the problem of finding regions of **TP**, specifically intersection volumes, which have an optimal value of the function $F(\phi, u, v)$. One approach to this is to simply sample points (ϕ, u, v) in **TP** to find one with an optimal value. An alternative perspective of this idea is that we pick a particular transformation and ask each match in parallel "Does this fall in your match-region?", for all possible matches. When we pick a transformation near the correct one the answer will be "Yes" for many matches. This is just what transformation sampling does.

A systematic approach would be to sample the entire parameter space on a regular grid, to find some such sample points which fall inside optimal intersection volumes. This is a finite but extremely large set of sample points. However, we can reduce the number of sample points considerably by noting that for a particular feature match we need only consider those sample points which actually fall inside its match region. Thus, assuming that an average of T sample points fall inside each match region, we need only consider $O(Tmn)$ transformation sample points, where mn is the number of feature matches.

4 The Recognition Algorithm

In this section we briefly outline the steps of the algorithm, and analyze its complexity. In a later section we will develop a more formal theoretical foundation for the algorithm and look at it in the context of parallel implementation.

4.1 The Algorithm

We will consider features consisting of straight line segments approximating the boundary contours. In its basic form the algorithm consists of the following:

- Extract image features
- Form all matches $< m, d >$ of model and image features
- Compute the range of valid relative rotations for each match
- Determine the set of rotation sample points in this range of rotations
- For each rotation sample point, compute the range of valid relative translations
- For each rotation sample point, determine the set of translation sample point in this range of translations
- For each complete parameter space sample point thus constructed, compute the value of $F(\phi, u, v)$ at the point
- Select the transformation sample points with optimal values of $F(\phi, u, v)$
- Determine the objects position in the image from the optimal transformation sample points

4.2 Complexity of the Algorithm

We will not consider the feature extraction phase of the algorithm. The complexity of the matching phase depends on how many transformation sample points we consider for each match, and on how many matches we have. Since for a particular match we are only interested in the transformations within its match region, how many transformation sample points does the match region enclose? We sample transformation parameter space on a regular grid in the 3 dimensions ϕ, u, and v. Let $\delta\phi$ radians be the sampling interval in the rotational dimension, and δt be the sampling interval in each of the translational dimensions. The *important point to note is that only the sample points which* fall within the match-region for some feature match need ever be considered. For a particular match the size of the match region depends on three things: R and Θ, the uncertainty in position and orientation respectively, and the difference in length between the model feature and the image feature. Thus, there are $r = \lfloor \frac{2\Theta}{\delta\phi} \rfloor$ rotation sample points to consider. See figure 2.

Considering again this particular feature match, for each valid rotation sample point, after the model segment has been rotated according to the specified rotation sample point, there will be a set of transformation sample points which fall in the range of translations, u and v, which bring the matched features within a distance of R. We call this set of sample points a *translation neighborhood*. For a giver rotation sample point, the width of the translation neighborhood in sample points, w, is given approximately by $w = \lfloor \frac{2R}{\delta t} \rfloor$. The length of this region is a function of the difference in length between the image and model segments. This is because the image segment, if shorter, can slide along the length of the model segment and still be within R. Let D_{ij} denote the difference in length between matched features $< d_i, m_j >$. For a given rotation, the length of the translation neighborhood in sample points is given approximately by $l_{ij} = \lfloor \frac{2R + D_{ij}}{\delta t} \rfloor$. So for match i, j, the number of discrete transformation sample points considered is $T_{ij} = rwl_{ij}$.

Suppose we have an image of a given size. If we consider many different models and data sets, we could determine reasonable average case values for l_{ij}. To simplify the analysis, let l be the maximum over many model and data sets, of the average of l_{ij} over the matches for each pairing of image and model feature sets. Defining $T = rwl$, the total number of different transformations that need be considered is $O(Tmn)$.

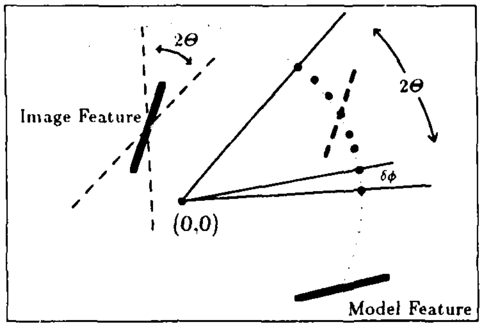


Figure 2: An illustration of the uncertainty bound Θ, and the set of possible rotation samples for a particular match.

Can we really consider the number of sample points per match region as a constant? The number of sample points, T depends on the sampling intervals $\delta\phi$ and δt, and on the difference in length between the matched features. The size of the image provides an upper bound on the difference in length between matched features, so this only depends on the size of the image, I. It remains to be shown in section 6 that $\delta\phi$ and δt are independent of m, n, and I. In this case T depends only on I, which might reasonably be considered constant.

5 Related Work

5.1 Transformation Sampling is not a Hough Transform

Clustering has been used as a complete recognition technique, for example see Stockman [7]. Transformation sampling is not simply a Hough transform or similar clustering technique. There are some key characteristics of the transformation sampling technique which make it much more effective than clustering techniques. Here we will argue that while the technique is similar to Hough transform techniques, there are some important differences.

5.1.1 Problems With Clustering as a Recognition Technique

The basic idea behind clustering techniques for recognition can be easily described procedurally. All pairs of model and image features are formed. For each such match, there is a relative transformation on the model feature, rotation and translation, that will align the matched features. This is computed, and a point is placed in the 3D transformation parameter space corresponding to this transformation. The idea is simply that correctly matched features will yield approximately the same relative transformation, and thus a cluster in parameter space will be formed. Good candidates for the correct transformation are found by searching parameter space for such a cluster. In principle, this is an excellent approach. In the ideal case there would be a sharp peak in parameter space corresponding to the correct transformation. In practice, there are two main factors which make correctly identifying the correct transformation difficult: error in the measurement of the pose of image features, and spurious points in parameter space due to incorrect matches.

First we consider the effect of uncertainty in the pose of the image feature. There is a strong coupling between the rotation component, and the translation component of the transformation aligning a model feature with an image feature. An error in the estimate of the orientation of an image feature results in an incorrect relative rotation, which in turn results in an incorrect relative translation. This has the effect of spreading out the cluster in space, so that although features are correctly matched, they do not all correspond to the same point in parameter space. If the feature is away from the center of rotation, it is easy to see that uncorrelated error in the orientation of several image features will lead to a helical shaped distribution of points in parameter space, for correctly matched features. There is another factor which contributes to the dilution of clusters: image feature fragmentation. When an image feature is only part of its corresponding model feature, there is no way a priori to determine a unique translation aligning the two features. Thus there is a region of uncertainty in which the point could fall in parameter space, further spreading out the cluster. One solution to the problem of spread out clusters is to smooth the parameter space. The idea is to represent each match in parameter space wherever it could fall, according to the pose uncertainty. It is difficult to smooth the space correctly, which results in matches falling in parameter space where they should not fall, and not falling where they should.

Second, we consider the points in parameter space contributed by incorrectly matched features. These points contribute to what might be called the overall *background noise* of the parameter space. As the level of this background noise increases, that is, as the number of points in parameter space due to incorrect matches increases, peaks or clusters in parameter space become harder to distinguish. The presence of unknown objects in the image results in more pairs of image and model feature, and more points contributed to parameter space, which contributes greatly to the background noise.

These clustering techniques are commonly implemented by tessellating parameter space into *bins*. All points falling within the same cell of this tesselation are considered members of the same bin. To find clusters we simply find the bin with the most points in it. To handle the effect of uncertainty in the pose of the image feature, the cells must be made large enough to include most of a cluster, and thus the size of the bins depends on the bounds on pose uncertainty. Alternativly, the tessalaiton is made smaller and the space is smoothed as above.

The bin with the most matches falling in it may not be associated with the optimal transformation, since it is quite possible that no single transformation will simultaneously align all matches in the bin. We can see this by noting that the regions of valid transformations for two different matches may intersect the same bin, but not intersect one another, creating a false peak in the tessalated parameter space. See figure 3. This is one reason why the clustering technique alone is not adequate for recognition. Grimson and Lozano-Pérez[8] simply use the Hough transform to roughly order the matches they will consider in a subsequent constraint based search procedure.

In summary, error in image feature pose, and fragmentation spread out the clusters in parameter space. Incorrect matches add to the background noise, watering down the relative strength of peaks in the space, and leading to false peaks. And finally, all features in a bin are not necessarily globally consistent.

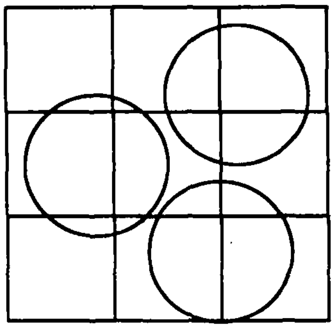


Figure 3: The squares represent slices of Hough bins, the circles represent regions of valid transformations for matches. Although a Hough bin may have several members, they are not necessarily mutually consistent.

5.1.2 The Advantages of Transformation Sampling

In contrast to clustering techniques, transformation sampling explicitly accounts for pose uncertainty and feature fragmentation.

Transformation sampling can be related to the Hough transform by considering the size of the cells to be infinitesimal, and the spacing of the cells in each dimension to be the transformation sampling intervals. Following the Hough transform analogy further, after each point is placed in parameter space it is smeared out so that it falls in all the bins that it possibly could given the pose uncertainty bounds. Thus every match is represented at all sample points at which it possibly falls in parameter space, and no more. Note that since it is as though the bin size were infinitesimal, all matches that fall in a particular bin are globally *consistent* with one another.

We noted that in a standard Hough transform technique, the presence of unknown objects in the image contribute to the background noise in the parameter space, and thus reduce the accuracy of the clustering technique. In the case of transformation sampling this background noise due to incorrect matches makes absolutely *no difference* in accuracy, but simply increases the complexity of the procedure since there are more model feature image feature matches to consider. Noise makes no difference since if an incorrect match is valid at a particular transformation sample point, then to the best of our knowledge it is a correct match.

Transformation sampling easily accounts for any amount of error in the image feature pose, by simply considering a larger number of possible transformation sample points. This mechanism also handles problems with image feature fragmentation since the fragment will be considered at all points along the model feature matched to it. Thus we are able to use all image data to the full extent, without limiting matches to be composed of features of essentially the same length.

5.1.3 False Positives versus False Negatives

We can characterize the main differences between transformation sampling and Hough transforms by noting that transformation sampling in a sense trades false negatives for false positives. We have argued that the Hough transform may indicate a popular transformation which in fact is not globally consistent. On the other hand, transformation sampling will never indicate anything but a globally consistent transformation. However, since we are sampling parameter space, it is possible that we could miss a peak in parameter space, and thus miss a transformation which the Hough transform may have found. It must be shown that given fine enough sampling intervals this will not happen. In section 6 we will examine this more fully.

5.2 Constrained Search Based Recognition

We have not considered in detail the use of a constrained search based recognition technique for parallel recognition because this formulation may not work well in parallel, and because it does not exploit all the available constraint. The systems by Grimson and Lozano-Pérez[8], and by Bolles and Cain[5] both rely on constrained search based upon the pairwise constraints between feature matches. The idea is to hypothesize a matching between model and image features from which a transformation can be derived. This is accomplished by finding a large set of pairwise consistent matches. We can see that this is a difficult problem by considering the feature matches as nodes in a graph, where there is an edge between nodes if the matches are pairwise consistent. This is simply the maximum clique problem on a graph. This problem is NP-Complete[12], but in practice this works well if the matches are considered in carefully chosen subsets to reduce complexity. However, it seems that this is not the best approach to consider for a parallel implementation for two reasons. First, it seems the sequential nature of the search process is what makes the constraints effective at pruning the search space. It is clear that we cannot simply explore the entire search space in parallel since it is enormous. Second, the pairwise consistency formulation does not exploit all the constraint available, since it looks for sets of *pairwise consistent* matches, when in fact any correct matching we construct must be *globally consistent*, that is, all matches must align under the same transform. The transformation sampling technique, on the other hand, is easily implemented in parallel, and explicitly relies on the global consistency constraint. Thus, transformation seems a more attractive approach to parallel recognition.

6 A Formal Basis for Transformat'on Sampling

One of the potential problems with transformation sampling is that an optimal transformation will be missed because the sampling intervals are two large. Due to the particular configuration of the measurement errors for the set of image features, it is possible that the optimal intersection volume, the place where most match regions overlap, is so small that no sample point falls inside it, thus missing it. In this section we illustrate that although we may miss the point of greatest overlap, it is very likely that we will sample a point nearby it in parameter space, where many of the correct match regions overlap.

6.1 Background

For this development we will consider point features, consisting of a position and orientation but without spatial extent. We will then extend the result to include the extended features used in the implementation. Each match between a model and image feature defines a region of transformation parameter space we call a match region. Transformation sampling seeks to find a region where many such match regions intersect, defining a transformation which accounts for a large amount of the data derived from the image. When dealing with point features, match regions have circular cross sections in planes of constant ϕ, we call the projection of these circular cross sections of match regions onto the $u - v$ plane *match circles*. In the $u - v$ plane, match circles follow circular paths as ϕ varies, of radius $|\vec{m}|$ centered at $\vec{d}$, given by $\vec{t}_{ij} = \vec{d}_j - M_\phi \vec{m}_i$ for feature match $< m_i, d_j >$.

Consider the projection of a slice of parameter space at constant $\phi = \phi_0$ onto the $u - v$ plane. Assume that ϕ_0 is the *correct* rotation we seek, aligning the orientations of the correctly matched model and image features. Let $\vec{t_0}$ represents the *correct* translation in $u - v$ plane. Then for all correctly matched model and image features, $|\vec{t_0} - \vec{t}| \leq R$ by definition of the position uncertainty. Graphically this means that the correct match circles include the point $\vec{t_0}$.

6.2 The Overlap of Match Regions

For each image feature, assume there is a probability density function characterizing the deviations of the measured feature position from the actual feature position. From this we can derive a probability density function $f_\rho(\rho)$ characterizing the distribution of $\rho = |\vec{t_0} - \vec{t}|$, the absolute distance in $u - v$ space between the correct translation and the translation calculated from the measured feature. In this analysis we will assume that $f_\rho(\rho)$ is zero outside $0 \leq \rho \leq R$. It is important to bear in mind that we are dealing with the case where the *correct* rotation, ϕ_0, has already been performed. Now consider a circle of radius $r = R - \rho$ centered at $\vec{t_0}$. See figure 4. This circle is completely contained within the match circle. Call this circle C_r. We can easily derive the probability distribution of the value r from $f_\rho(\rho)$: $f_r(r) = f_\rho(R - r)$. Lastly consider the probability P_{r_0} that for a particular feature match, the radius of the little circle, C_r, is less than some constant r_0:

$$P_{r_0} = \int_0^{r_0} f_r(r)$$

We will show that a large number of match circles overlap a circle C_{r_0} of some radius r_0, and thus we will find these by sampling points which fall inside C_{r_0}.

We now calculate the probability that more that n_0 feature matches each produce a circle C_r with radius $r < r_0$. Assume there are N correctly matched image and model features. That is, there are N of the model features visible in the image. Assuming the measurement errors are independent, the probability that n of these N matches each produce a small circle, C_r with $r \leq r_0$ is given by the binomial distribution:

$$P_n(n) = \binom{N}{n} P_{r_0}^n (1 - P_{r_0})^{N-n}$$

so the probability that more that n_0 feature matches each produce circles C_r with radius $r \leq r_0$ is then

$$P_{>n_0} = 1 - \sum_{i=0}^{n_0} P_i(i) = 1 - \sum_{i=0}^{n_0} \binom{N}{i} P_{r_0}^i (1 - P_{r_0})^{N-i}$$

6.3 Sampling Parameter Space

If $P_{>n_0}$ is small for small values of n_0, then it is likely that most correct match circles mutually intersect over a region C_r of radius

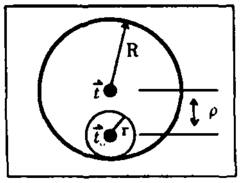


Figure 4: A circle of radius r centered at $\vec{t_0}$ fits inside the match circle.

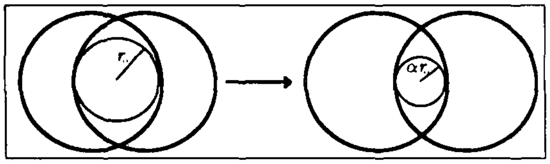


Figure 5: As ϕ moves away from the correct value ϕ_0, the circle C_r of radius r_0 centered at $\vec{t_0}$ shrinks to radius αr_0.

at least r_0. If we choose the translational sampling interval δt according to r_0, we can ensure that a sample point will fall in the circle C_r. Again, we have assumed that rotation by the correct amount, ϕ_0, has been performed. We now consider the structure of parameter space as ϕ moves away from the correct value $\phi = \phi_0$. The velocity at which a match circle moves in the $u - v$ plane as ϕ varies is proportional to $|\vec{m}|$, the distance of its center from the center of rotation.

We can bound this velocity as follows: If the maximum dimension of the image is I, then we can shift the model features such that $|\vec{m}| \leq \frac{\sqrt{2}I}{2}$ for all model features. In this case the maximum velocity with respect to ϕ of a match circle is $\frac{\sqrt{2}I}{2}$ rad^{-1}. Thus, the circle C_r of radius r_0 which probably intersects at least $N - n_0$ match regions can shrink in radius no faster than the match circles forming its boundary can move.

6.3.1 Sampling Intervals

Say we are willing to let C_r shrink to radius αr_0 where $0 < \alpha \leq 1$. In this case, as we vary ϕ from ϕ_0, as long as

$$\delta\phi = |\phi - \phi_0| \leq \frac{2r_0(1-\alpha)}{\sqrt{2}I} = \frac{\sqrt{2}r_0(1-\alpha)}{I}$$

C_r will have radius at least αr_0. See figure 5.

This allows us to define the rotation sampling interval $\delta\phi$ and the translation sampling interval δt, in terms of r_0 and α. For in order to be sure we sample within a circle of radius αr_0 we must have

$$\delta t \leq \frac{2\alpha r_0}{\sqrt{2}} = \sqrt{2}\alpha r_0$$

7 Discussion

There are some strong assumptions we are making in this analysis. First, that the deviation of the measured position of an image feature is randomly distributed, and that the magnitude of this deviation is characterized by $f_\rho(\rho)$. Second, that these events are independent of each other for different image features. It is possible that the error in position measurement is systematic, or that the error in measurement for one feature is correlated with that of another. Note, however, that the dependence on the actual form of the distribution $f_r(r)$ is minimized since we deal with its integral P_{r_0}, thus we need only consider large classes of functions $f_r(r)$.

Nonetheless, we can view this analysis as a counting argument. The probabalistic analysis facilitates characterization of the space of all possible error configurations, where a point in this space is given by a set of position errors, one for each image feature correctly matched. The probabalistic analysis determines the fraction of all possible error configurations for which we can be certain not to miss the event in parameter space we seek when sampling it. Note that this analysis provides a very loose lower bound on δt and $\delta\phi$ since we assumed the worst possible behavior of the circle C_{r_0}.

7.1 An Example

Assume that the measured position of an image feature is equally likely anywhere within a distance R of the true position. The probability that the measured features falls within a distance ρ_0 of the true position is

$$P_{\rho < \rho_0} = \int_0^{\rho_0} f_\rho(\rho) = \frac{\pi\rho_0^2}{\pi R^2}$$

so

$$f_\rho(\rho) = \frac{2\rho}{R^2}$$

thus

$$P_{r_0} = \int_0^{r_0} f_\rho(R - r)dr = \frac{r_0(2R - r_0)}{R^2}$$

Let $n_0 = \delta N$, $r_0 = \beta R$ and α be as above where $0 \leq \alpha, \beta, \delta \leq 1$.

As a specific example, let $\delta = \frac{1}{2}$, $\beta = .159$, and $N = 30$, so $n_0 = \delta N = 15$. Let $R = 5$, so $r_0 = \beta R = .795$. Thus $P_{r_0} = \beta(2 - \beta) = .293$. Finally, $P_{\geq n_0} = .005$.

Here we see that over 99% of all possible error configurations result in a circle C_r of radius $r_0 = .795$ which is contained in more than $\frac{N}{2}$ match circles, at the correct rotation.

Say the image size, I, is 256. Picking $\alpha = \frac{2}{3}$ to minimize the density of sample points in parameter space we have $\delta\phi = .084$ degrees, and $\delta t = .75$

These numbers result from the assumption that the measured positions of image features are uniformly distributed, in which case the probabalistic analysis is analogous to a counting argument. Note that this is an extremely unfavorable distribution $f_\rho(\rho)$. We would expect that the probability is high that ρ is small. Thus with a more reasonable distribution we expect larger sampling intervals. As an example of a more likely distribution of the position error, assume that $f_\rho(\rho)$ is a truncated half-gaussian with standard-deviation $\sigma_\rho = \frac{R}{2}$, with $f_\rho(\rho) = 0$ outside $0 \leq \rho \leq R$ and

$$f_\rho(\rho) \propto e^{-\frac{\rho^2}{2\sigma_\rho^2}}$$

inside the interval. In this case $P_{\geq n_0} = .013$, $\delta t = 2.5$, and $\delta\phi = .28$ degrees, where we used $\delta = \frac{1}{2}$, $\beta = .535$, $\alpha = \frac{2}{3}$, and $I = 256$.

7.2 Extended Features

The argument for point features also works for extended features such as line segments. For a particular feature match at a particular rotation there is a region of possible translations. The length of this region is a function of the position uncertainty and the difference in the lengths of the model and image features. An equivalent situation is given when we subtract the length of the image feature from both the model and the image features. Assuming the image feature is less than or equal in length to the model feature, we then have an image feature which is a point paired with a model feature which is either a point or a line. It is easy to see that in this case the amount of overlap of match regions is equal to or greater than the case where only point features are involved. Thus, the point feature analysis provides a lower bound on the performance of the technique when using extended features.

8 Implementation

In this section we discuss the implementation of this recognition procedure for a particular model of parallel computation. Because of the local nature of many of the computations, the simplicity of the few global computations required, and the modest processor requirements, this recognition procedure is efficiently implemented on a parallel architecture.

8.1 Background

The model of parallel computation we use is the Exclusive Read Exclusive Write (EREW) Parallel Random Access Machine (PRAM) model[15]. We restrict the PRAM to Single Instruction Multiple Data (SIMD) since we do not require the more general Multiple Instruction Multiple Data (MIMD) model.

Before we discuss the specific aspects of the algorithm we need to be familiar with some basic parallel operation. Two of what we might call primitive parallel operations are scan operations and permutation (parallel exclusive read and write). There are a number of simple operations, constructed with primitive operations, that are useful such as sorting. Scan operations, sorting, and parallel reads and writes will play a key part in the implementation of this recognition technique.

Conceptually we might view the parallel machine as linear vector of processors and memory[16]. Permutation of data elements in this vector is equivalent to exclusive *read* and *write* operations. In this model these operations take $O(1)$ time. Probably less familiar to the reader are the *scan* operations, which are based on the application of binary associative operators over a set of values. Strictly speaking, in the EREW model the scan operations can be done in $O(\lg M)$ time on M processors, although the scan operation could be considered a primitive operation requiring $O(1)$ time[3]. As an example, the scan operation can be used to distribute a value from one processor to several others, and to compute the cumulative sum, maximum, and minimum of values represented in a set of processors. The scan is used extensively in the implementation of this recognition procedure. Finally, sorting of values stored in a set of processors can be accomplished in $O(\lg^2 M)$ time for M values[4]. These operations are the basic tools with which we build the parallel recognition procedure.

8.2 The Parallel Nature of This Formulation

All of the operations required to implement this procedure are fast in parallel. Furthermore, the number of processors required is linear in mn, the number of feature matches. The fundamental idea of the parallel implementation is that each processor manipulates one data structure. For example, throughout most of the computations, each processor has a representation of one model feature, and one data feature, which form a particular match. This processor will perform operations on these features such as finding the difference in their lengths, rotating and translating the model feature, etc. These are all simple $O(1)$ time operations, and are performed on all processors at once.

In section 4.1 we presented an outline of the algorithm. In this section we will consider the implementation of these steps on the EREW model.

- Form all matches $< m, d >$ of model and image features

Starting with each model feature represented one per processor, and each image feature one per processor, forming all pairs of model and image features, $\{m_j\} \times \{d_i\}$, requires $O(\lg mn)$ time, and is accomplished using write and scan operations by replicating each image feature m times, and each model feature n times and reordering the features so all matches are formed, one per processor.

- Compute the range of valid relative rotations for each match
- Determine the set of rotation sample points in this range of rotations
- For each rotation sample point, compute the range of valid relative translations
- For each rotation sample point, determine the set of translation sample point in this range of translations

This is done by spreading the work over several processors. First, for each feature match, the set of valid rotation sample points is calculated, and the match is replicated in a new set of processors, one copy per rotation sample point, forming a set of *rotated matches*. Then, each new processor representing a rotated match actually rotates the model feature according to the rotation sample point, and calculates the set of translation sample points for its particular rotation. The rotated matches are then replicated, one copy per translation sample point per processor.

The result is that each original match $< m_i, d_j >$ is represented in T_{ij} seperate processors, one per transformation sample point where T_{ij} is the number of transformation sample points falling in its match region. The processes of calculating these sets requires $O(1)$ time, and distributing them one per processor requires $O(\lg(Tmn))$ time.

- For each complete parameter space sample point thus constructed, compute the value of $F(\phi, u, v)$ at the point

Calculating $f_{ij}(\phi, u, v)$ for each match takes $O(1)$ time. To calculate $F(\phi, u, v)$ in the case were this consists of the sum of the $f_{ij}(\phi, u, v)$ requires first sorting the matches according to their associated transformation sample points, and arranging them in contiguous groups of processors with the same transformation sample point. $F(\phi, u, v)$, for each transformation sample point, is computed by performing a *plus scan* of the $f_{ij}(\phi, u, v)$ for matches instanciated at the same transformation sample point. The sort and scan require, respectively, $O(\lg^2(Tmn))$ and $O(\lg(Tmn))$ time.

- Select the transformation sample points with optimal values of $F(\phi, u, v)$

This final step is done by performing a simple global maximum operation on the values of $F(\phi, u, v)$ for each transformation sample point (ϕ, u, v). The optimal value of $F(\phi, u, v)$ can be found in parallel in $O(\lg(Tmn))$ time, giving the best transformation.

It is interesting to note that the final operations are similar to a histogramming procedure where rather than explicitly representing each bin, we sort the items to be histogrammed according to the bin they fall into, and use a scan to determine the number that fell in each occupied bin. Only the occupied bins are represented, so the number of possible bins can be very large, as long as only a reasonable number are ever occupied.

Thus we see how the recognition procedure is implemented. The procedure requires $O(\lg^2(Tmn))$ time and $O(Tmn)$ processors.

8.2.1 Pre-Pruning of Parameter Space

Although this method requires $O(Tmn)$ processors, in practice T can be large. For example, in figure 1 $T = 78$ while $m = 42$ and $n = 269$. Thus in practice, we might have 10^6 transformation sample points to consider, with only 64K processors to use. One approach is to simply iterate over sets of matches and transformation sample points, using all available processors at each iteration.

We employ two absolute thresholds to eliminate unlikely transformations: a minimum on the fraction of the model perimeter explained by the data, and a minimum on the number of feature matches valid at a transformation sample point. The idea is that there are many regions of transformation parameter space in which only a few matches fall.

To implement this, a coarse translational Hough transform is computed for each match, at each of its associated rotation sample oints. This is much faster than full translation sampling and allows us to eliminate unpopular transformations before we consider them further. For example in figure 1, there were over 12,000 rotated-matches falling alone in large regions of translation space. Just eliminating these allows us to disregard $wl \times 12,000 \approx 150,000$ translation sample points, where wl is the number of translation sample points for a given rotation sample point. We can be sure that under the criterion that a minimum amount of the model must be visible that we are not eliminating any correct hypotheses.

9 Experiments

9.1 Connection Machine Implementation

The system implemented is a simple demonstration of this method. For simplicity in the initial implementation, only a single instance of a single model is present in the image. In the next section we will show why the algorithm works with multiple instances of a single model as well as a small number of different models with multiple instances.

We define $f_{ij}(\phi, u, v)$ **length**(d_j) inside the match region and 0 outside. The best transformation is chosen as that with the maximum value of $F(\phi, u, v) = \sum_{i,j} f_{ij}(\phi, u, v)$ without further reasoning. We would not build a complete recognition algorithm in this way, but this serves to demonstrate the idea.

The Connection Machine [13], or CM-1, is a parallel computer based on a hypercube network, which simulates reads and writes on an M processor EREW machine in $O(\lg M)$ time with high probability. Each processor has approximately 4K bits of local memory, and is capable of general computation.

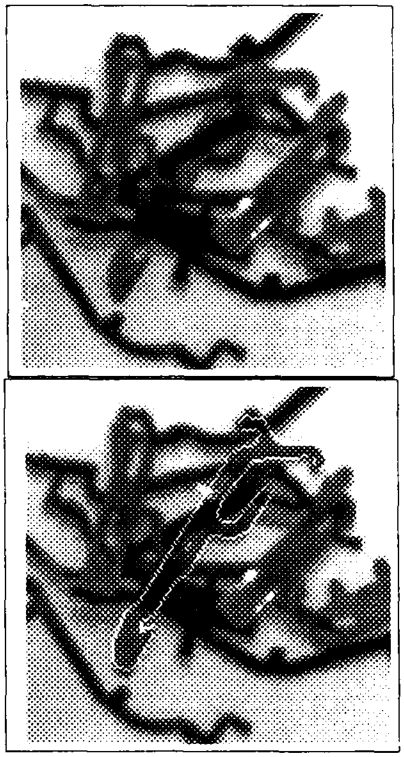


Figure 6: n 41, m 241, $\delta\phi = 2$ deg, $\delta t = 2$ pixels, $\Theta = 6$ deg, $R = 5$ pixels, T 144

The interface to the CM-1 is through a Symbolics 3600 series computer, to which special imaging hardware is attached facilitating the loading of image data into the CM-1. The entire recognition procedure consists of loading the data image into the CM-1, extracting intensity edges, forming a polygonal approximation of these edges to form image features, and then matching these against previously stored model features.

A parallel implementation of the edge detector developed by Canny[2][1] is used to extract intensity edges. The polygonal approximation technique employed also exploits parallelism, but is an extremely simple technique consisting of recursive splitting of straight line approximations to the edge curves until some error bound is achieved.

The system was implemented on a quarter-sized CM-1, consisting of 16K processors. The full machine has 64K processor, however the quarter machine can simulate the full machine with very close to a factor of 4 slowdown. Thus we assume that we have implemented this procedure on a 64K processor machine.

9.2 Results

The initial implementation has been run on very heavily occluded input scenes. Examples of typical images are shown in figure 6, figure 7, and figure 8. The values of the parameters used for these examples are shown. The implementation has not been

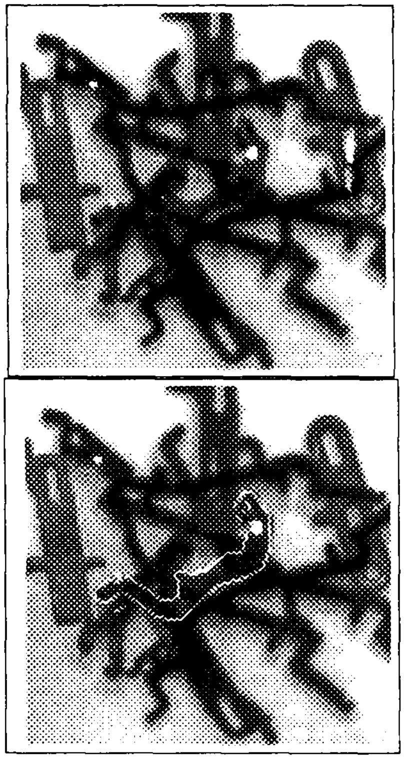


Figure 7: $n = 31$, $m = 259$, $\delta\phi = 2$ deg, $\delta t = 2$ pixels, $\Theta = 6$ deg, $R = 5$ pixels, $T = 121$

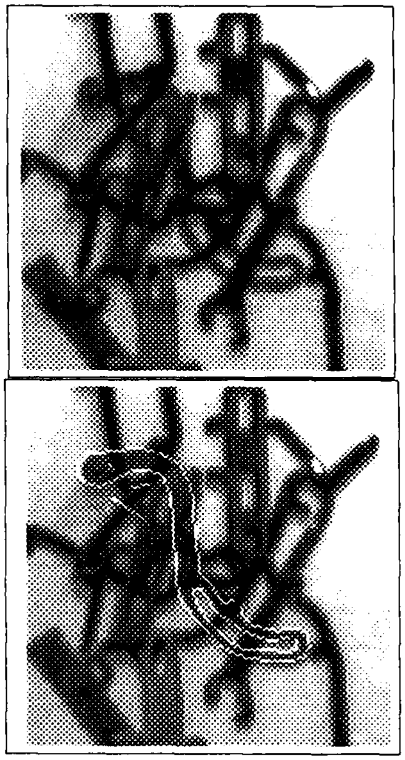


Figure 8: $n = 46$, $m = 225$, $\delta\phi = 2$ deg, $\delta t = 2$ pixels, $\Theta = 6$ deg, $R = 5$ pixels, $T = 146$

optimized for speed, and there is room for speedup. However, the transformation sampling portion of the algorithm runs approximately 5 seconds on a 16K processor machine, or approximately 1 second on a full 64K processor machine. The Canny edge detector runs in approximately .3 seconds, and the polygonal approximation procedure runs in about 2 seconds on a 16K machine.

10 Extensions

10.1 Multiple Models, Multiple Instances

The algorithm and the resulting recognition system is more general than we have initially described. The algorithm can recognize multiple instance of an object in the image. To see this, note that after the transformation sampling procedure we have available hundreds of different hypothesized transformations, all of which are globally consistent, valid interpretations of the image data. To find multiple instances of a model, all the work is already done and all we need to do is pick those other transformations which meet the minimal criteria.

Since the model is just a collection of features, we could simultaneously recognize 2 or more objects by simply including the new model features with the original set. All that is necessary is that we keep the calculations for each different model separate from one another. This is trivial to do in this impiementation.

10.2 Refining the Transformation

For demonstration purposes, in this implementation we have relied on one stage of transformation sampling for the entire recognition engine. This provides a good approximation to the best possible transformation. A two stage procedure could be used to refine these transformations. In the first stage we sample space as coarsely as possible provided that we can be sure to find approximately the correct feature mapping and associated transformation. We can then refine a small number of hypothesized transformations further. The advantage of this multi-stage approach is reduction in computational complexity; the number of sample points that need be initially considered is smaller.

The refinement stage could be a second iteration of the transformation sampling technique concentrating only on a finer grid of sample points within one interval of the original sampling grid, and near a hypothesized transformation from the first stage. Another possibility is a parallel gradient based search of the transformation parameter space, utilizing appropriate definitions of the functions $F(\phi, u, v)$, and $f_{ij}(\phi, u, v)$ which would produce a smooth and continuous search space.

10.3 Model Libraries and 3D Recognition

This robust recognition technique was not intended to handle large libraries, but rather to be able to recognize a small number of possible objects. The system would, however, provide a second stage for robust verification, following an indexing first stage

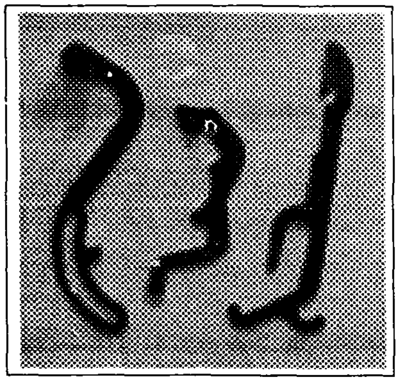


Figure 9: The models.

which selects a small set of candidate models from a much larger library of models.

Lastly we consider extensions to 3D recognition from 3D range data. We cannot directly apply this procedure to 3D feature matches since there is no unique transformation aligning two lines in 3D. We can, however, introduce constraint by considering pairs of model features with pairs of image features, and then use transformation sampling as before, except in a 6D parameter space. The complexity of the procedure rises dramatically in this case due to the additional degrees of freedom, however in principle the technique will work.

11 Conclusion

We have introduced transformation sampling as a technique for 2D model-based recognition, and given a formal basis for the technique and the determination of the appropriate sampling intervals. The technique offers several advantages to existing techniques based on the Hough transform.

One of the main goals was to produce a recognition algorithm suitable for parallel implementation. The algorithm developed not only provides an effective recognition technique, but is highly parallel in nature.

We have described and demonstrated an implementation of a recognition system for large SIMD parallel computers. The processor requirements are $O(Tmn)$, linear in the number of pairs of model and data features, and the time requirements $O(\lg^2(Tmn))$. The system is capable of determining a very close estimate of the transformation which aligns a model with its instance in an image.

Acknowledgments

This report describes research done at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology, and was funded in part by an Office of Naval Research University Initiative grant under contract N00014.86K-0685, in part by the Advanced Research Projects Agency of the Department of Defense under contract N00014.85K-0124, and Army contract DACA 76-85-C.0010.

The author would like to thank several of the members of the MIT vision group for many interesting and fun conversations, especially Eric Grimson, David Jacobs, and Jim Little who read earlier drafts of this paper.

References

[1] J. Little, G. Blelloch, T. Cass 'Parallel Algorithms for Computer Vision on the Connection Machine' Proc. of the First Int. Conf. on Computer Vision, 1987.

[2] J. F. Canny, 'A Computational Approach to Edge Detection', IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol 8, no. 6, Nov. 1986.

[3] G. Blelloch, 'Scans as Primitive Parallel Operations' Proc. Int. Conf. on Parallel Processing, 1987.

[4] T. Leighton, 'Tight Bounds on the Complexity of Parallel Sorting' IEEE Transactions on Computers, Vol C-34, no. 4, April 1985. Proc. Int. Conf. on Parallel Processing, 1987.

[5] R.C. Bolles and R.A. Cain, 'Recognizing and Locating Partially Visible Objects, the Local-Feature-Focus Method', Int. J. Robotics Res., Vol 1, no. 3, 1982.

[6] N. Ayache and O. D. Faugeras, 'HYPER: A New Approach for the Recognition and Positioning of Two-Dimensional Objects', IEEE Trans. on Pattern Analysis and Mach. Intelligence, Vol. PAMI-8, no. 1, January 1986.

[7] G. Stockman, S. Kopstein, and S Bennet, 'Matching Images to Models for Registration and Object Detection via Clustering', IEEE Trans. on Pattern Analysis and Mach. Intelligence, Vol. PAMI-4, no. 3, May 1982.

[8] W.E.L. Grimson and T. Lozano-Pérez, 'Model-based recognition and localization from sparse range or tactile data', Int. J. Robotics Res., Vol 3, no. 3, 1984.

[9] J.L. Turney, T. N. Mudge, and R. A. Volz, 'Recognizing Partially Occluded Parts', IEEE Trans. on Pattern Analysis and Mach. Intelligence, Vol. PAMI-7, no. 4, July 1985.

[10] R. O. Duda and P. E. Hart, 'Use of the Hough Transformation to Detect Lines and Curves in Pictures', Communtications of the ACM Vol 15 no. 1, January 1972

[11] D.H. Ballard, 'Generalizing the Hough Transform to detect Arbitrary Shapes', Pattern Recognition Vol 13, no. 2, 1981

[12] M.R. Garey and D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Freeman, San Francisco. 1979

[13] W. D. Hillis, *The Connection Machine*, MIT Press, Cambridge, 1985.

[14] S.A. Cook 'A Taxonomy of Problems with Fast Parallel Algorithms', Information and Control 64, 1985.

[15] S. Fortune and J. Wyllie, 'Parallelism in Random Access Machines', Proc. ACM Symposium on Theory of Computing', 1978.

[16] G. Blelloch, Forthcoming PhD Thesis, MIT Artificial Inteligence Lab, June 1988.

The Concept of an Effective Viewpoint [1]

J.L. Mundy, A.J. Heller and D.W. Thompson

General Electric Corporate Research and Development Center
Schenectady, NY 12301

Abstract

The power of object recognition features as a function of viewpoint is explored. A prior system for model-based recognition is examined, using the geometries of the relevant match feature to define a viewpoint based error function. General rules for determining degenerate viewpoints of a feature are proposed. The uses of the function in selecting promising match features is discussed. Finally, results of the recognition system are presented along with an analysis of the relevant error values, demonstrating the ability of the error metric to predict the reliability of features at arbitrary viewpoints and to derive a minimal set of effective features.

1 Introduction

1.1 Previous Work

The area of model-based vision is receiving increased acceptance as a practical approach to the recognition of objects in unconstrained intensity images [Lowe], [Ayache and Faugeras], [Huttenlocher and Ulmann], [Grimson and Lozano-Perez], [Ikeuchi], [Thompson and Mundy]. The central mechanism is the use of a three dimensional geometric model to constrain the analysis of a two dimensional intensity image projection. The recognition of an object involves the determination of an appropriate match between the model and its projection. The consistency of the projection transformation is the main criterion for the validity of feature assignments between the object model and the segmentation of the intensity projection.

The image data is segmented into boundary curves and regions which are then represented as geometric entities. A grouping step is required to form a sufficient number of constraints to determine the transformation between the object model and the image. For example, Huttenlocker uses three image point correspondences to establish the affine transformation between the model and the image coordinate frames. Lowe uses grouping of parallel lines and significant edge junctions to index feasible model assignments. Grimson and Lozano-Perez as well as Ayache and Faugeras use a cost-directed sequential search to establish consistent model assignments for edge and vertex features. Ikeuchi uses characteristic views [Chakravarty] to index into a decision tree that is automatically generated from a solid object model. The characteristic view features are defined in terms of unique region shapes and junction configurations.

An overriding theme of all of this work is that parameters of geometric features which are extracted from the image data support the computation of the transformation between the image and model coordinate frames. It thus becomes an important issue to examine the errors associated with this computation and the resulting effect on the validity of feature matches and consequent object recognition performance. Ikeuchi and Kanade have already begun to explore the effectiveness of various types of sensors in terms of viewing geometry and feature strength [Ikeuchi and Kanade].

We will pursue this question in the context of a particular model matching approach which we have been developing over the past several years. Our approach is based on the observation that the affine projection transformation can be determined from just two vertices and the edge directions associated with one of the vertices [Thompson and Mundy].

1.2 The Vertex-Pair Feature

The vertex-pair geometry is shown in Figure 1. The line segment connecting the two vertices is called the *spine*. This segment merely indicates the grouping of the two vertices and does not actually have to exist in the segmentation or the model. We choose one of the vertices to be the base vertex and define the angles between two edges intersecting at the vertex and the spine as α and β.

The main advantage of this image feature is that only two image vertices need to be grouped in order to establish the full six parameter transformation. The edges associated with one of the vertices are naturally available since image vertices are defined in terms of the intersection of boundary edge segments. The pair-wise vertex grouping leads to an algorithm which has complexity proportional to $M(N-1)^2$ where N is the number of image segmentation vertices, and M is the number of vertex-pairs defined in the model.

[1] This work was supported in part by the DARPA Strategic Computing Vision Program in conjunction with the Army Engineer Topographic Laboratories under Contract No. DACA76-86-C-0007.

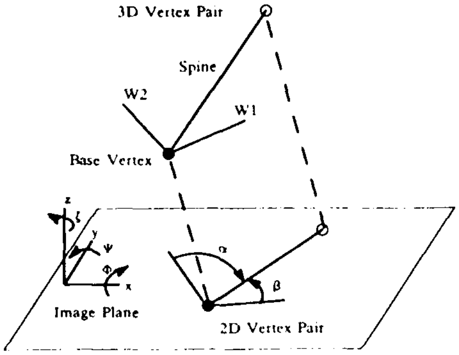


Figure 1: The three dimensional vertex-pair and its affine projection.

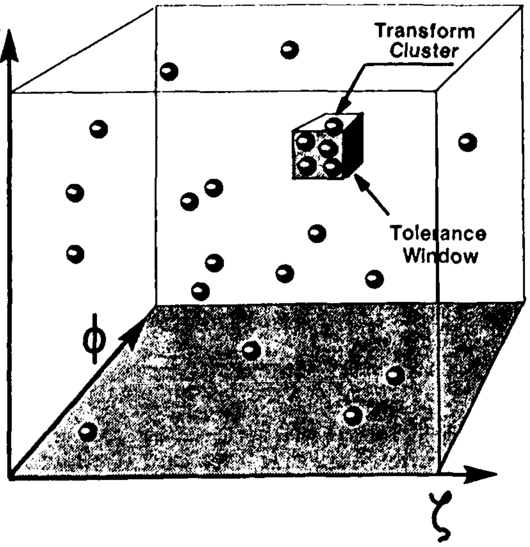


Figure 2: The clustering of transform values to establish valid model feature assignments. The figure shows the distribution of transform values in a three dimensional subset of the six dimensional affine transform space.

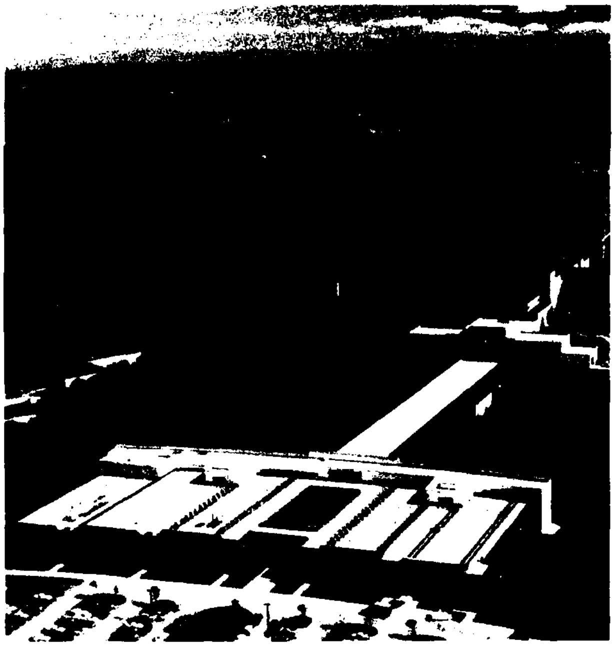


Figure 3: A view of the GE Research and Development Center.

More complex features become exceedingly expensive; the cost increases exponentially with the number of primitive features in the group. We also observe that the fragility of current segmentation algorithms favors approaches that do not require the reliable extraction of complex features such as trihedral junctions or well formed object face projections.

In our approach, all of the possible assignments are made *between the object model vertex-pairs* and the feasible vertex-pairs in the image segmentation. This exhaustive approach is based on the view that there is no reliable way of throwing out assignments outside of the context of the transform coherence that is generated by the matching process itself. However, we do eliminate vertices that do not have incident edges that are long enough to establish accurate projection parameters, in order to reduce the computational complexity as much as possible. In current experiments, we typically consider edges shorter than 5-10 pixels to be too short to provide an accurate transformation.

Each vertex-pair assignment leads to a transformation between the model and its proposed image projection. In our approach, these transformations are clustered in the transform parameter space to determine valid assignments. This validity is based on the concept of viewpoint consistency [Lowe], which is the observation that all points of an object will project into an image with the same transformation mapping function. Thus, if one has a correct set of assignments, then the transformations computed from those assignments, should all agree. The clustering concept is illustrated schematically in Figure 2.

We have carried out a number of experimenis in using the vertex-pair maiching system to recognize objects in outdoor scenes, including some experiments with infra-red data from the April 1985 U.S. air raid on Libya which we discuss later. In general, the results have been quite encouraging in terms of the ability of the algorithm to deal with clutter and occlusion.

An example of this matching process is given for the image of the GE Research and Developme.. Center in Figure 3. T'· segmentation for this image is shown in Figure 4. The model for a wing of the R&D Center is shown in Figure 5. The match between this model and the image data is shown in Figure 6.

These experiments have indicated the need for further developments to account for the errors associated with the placement of features in the image segmentation and the effect of

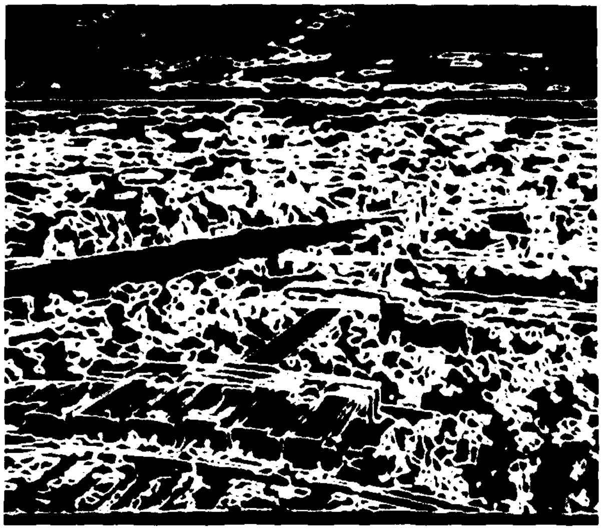


Figure 4: The segmentation into edges and vertices.

Figure 5: A model of the west wing.

these errors on clustering. We now consider these issues in more detail.

1.3 Weaknesses of the Current Approach

It is clear that the success of the clustering process is crucially dependent on the accuracy with which the transformation can be determined from each vertex-pair assignment; we depend on having a small cluster radius threshold in transform space to filter out incorrect matches.

In our algorithm we depend on accurate locations for the vertices and accurate angles between the direction of the spine of the vertex-pair and the other two edge directions. For some viewpoints, the uncertainty in these feature parameters can be amplified in the transformation parameters which give the three dimensional location and orientation of the object model with respect to the image coordinate frame.

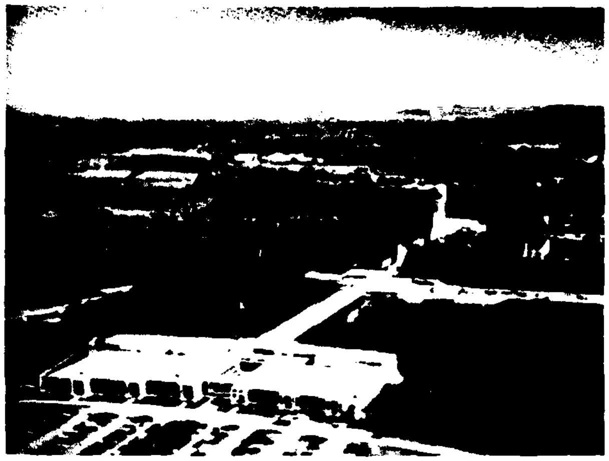


Figure 6: The resulting match of the model determined by the vertex-pair algorithm.

The current implementation attempts to provide for this uncertainty in the transform values by including all transform solutions which would correspond to an assumed spread in the image feature parameters. For example, we assume that the projected angles, α and β, can be in error by 5^0 due to uncertainty in the image segmentation process. This propagated uncertainty in the model transformation can lead to a range of solutions for the model match. Each solution corresponds to a slightly different value for the rotation parameters of the model transformation.

The relationship between image segmentation errors and the model transformation uncertainty will, of course, depend on the viewpoint. It is also the case that degeneracies will exist in the viewing projection. In particular transformation equations become degenerate for viewpoints which are collinear with the edges and spine of the vertex-pair.

In the current implementation, the model vertex-pairs are manually selected with an interactive graphics editor which operates on the object model. There is no guarantee that the vertex-pairs selected in this manner will have a good error performance. That is, it is essential that a set of vertex-pairs be selected which provide as accurate an estimate as possible of the model transform parameters, over all viewpoints. It is thus desirable that we extend the implementation as follows:

- Establish an error metric for the model transformation which can serve as a cost function in optimizing the selection of model vertex-pair features.
- Clarify the nature of viewing degeneracies for the vertex-pair feature, so that these viewpoints can be avoided in the clustering process.
- Implement an optimization algorithm to automatically select vertex-pairs.

The remainder of the paper is devoted to a discussion of these issues and will include some experimental results obtained for an error metric which appears to satisfy the criteria we just stated.

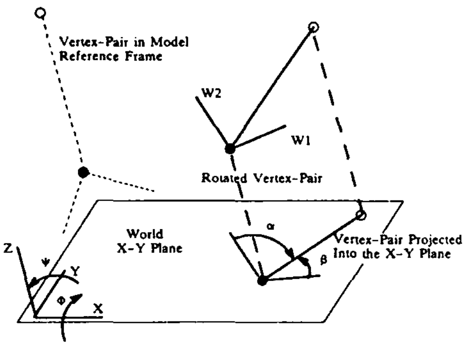


Figure 7: The tip-tilt transformation of the vertex-pair with respect to the x-y plane.

2 The Affine Image Transformation

2.1 Computing the Transformation

In a previous reference [Thompson and Mundy], we demonstrated that the affine transformation is an effective approximation to the general perspective case. In summary, the affine tranformation is characterized by an orthographic projection with a scale factor. Thus the full effects of rotation and viewing distance are represented in the affine transform. A consequence of the affine approximation is that parallel lines are projected as parallel lines. Thus, the usual perspective phenomenon of the vanishing point for parallel lines is not predicted by the affine approximation. However, for objects which are compact, compared to the viewing distance this perspective distortion is not large.

To clarify the following discussion, we briefly review the computation of the affine transformation parameters from the vertex-pair and its projection in the image viewplane. First, the transformation is defined in terms of rotation and translation parameters. The three dimensional rotation is defined in terms of successive rotations about the x,y,z coordinate axes. We refer to these angles as ϕ, ψ, ζ. The translation parameters are x,y and s, where we take the x-y plane to correspond to the image plane. The scale factor, s, is the ratio of the size of the object in the image to the actual projection of the object model onto the x-y plane. This scale factor is inversely proportional to the distance from the object to the camera.

The most important step in determining the transformation is determining the tip and tilt angles of the model with respect to the image plane, ϕ and ψ. We have shown that these angles can be determined directly from the projected angles between the vertex edges and the spine, α and β. These angles are defined in Figure 1.

In our current implementation, we precompute the functional relationships, $\phi(\alpha, \beta)$ and $\psi(\alpha, \beta)$, and store them as a look-up table computed for each vertex-pair defined in the model. The values of (α, β) are computed when an image vertex-pair is associated with a model vertex-pair and then the table is used to retrieve (ϕ, ψ). The model vertex-pair is then transformed by the (ϕ, ψ) rotations about the x and y axes to establish its projection into the x-y plane. This step is illustrated in Figure 7. .

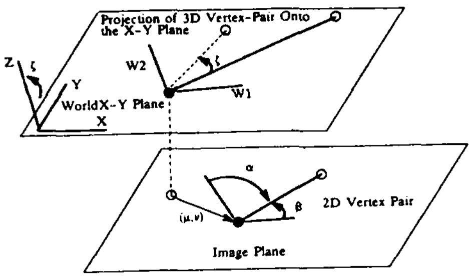


Figure 8: The determination of ζ and x-y translation.

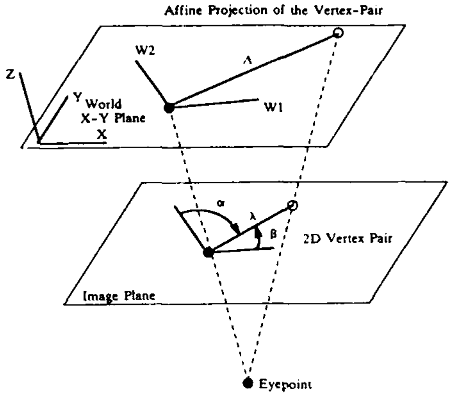


Figure 9: The detcrmination of the projection scale factor.

The remaining rotation angle, ζ, can be determined by observing the angle between the projected model spine and the actual spine direction in the image segmentation. Since the tip and tilt angles have been removed, this angle corresponds directly to ζ. This relationship is shown in Figure 8.

The translations in the x-y plane are also determined as shown in Figure 8. These translation components are simply computed from the translation vector, (μ, ν), between the projected model base vertex and the actual base vertex in the segmentation.

The final translation parameter is given by the ratio of the lengths of the projected model spine and the actual length of the spine in the image. This relationship is illustrated in Figure 9.

In the discussion to follow we focus on the determination of (ϕ, ψ). The error sensitivity of the other parameters of the transformation depends mainly on foreshortening and scale which are also functions of (ϕ, ψ). The effects of viewing distance and viewing orientation are thus mixed, which leads to the idea of multi-resolution model feature sets. We will consider this issue in a later section.

The key issue is how the vertex-pair is projected as the viewpoint moves over the surface of the standard *viewsphere*, which is a spherical surface defined to represent the possible viewing orientations of the image plane with respect to the object. The object is considered to be at the origin of the sphere and the points on the surface of the sphere define a vector orientation from a given point to the origin. This vector corresponds to a possible viewpoint orientation. A rotation about this viewing axis will not affect the nature of the projection of the model into to the image plane, since the image plane coordinate axes can be redefined to compensate. In our approach to assignment clustering, we find it convenient to represent three dimensional rotations in terms of sequential rotations about the axes of the camera coordinate system. However, we can relate any location on the viewing sphere to the (ϕ, ψ) rotational parameter subspace.

The optimization problem thus becomes one of selecting vertex-pairs so that the viewing sphere is exhaustively covered. That is, for each point on the viewing sphere, there should be an adequate number of potentially visible vertex-pairs which have acceptable error performance and are not degenerately viewed for that viewing orientation. Naturally, one can not guarantee that a given vertex-pair will be visible, since it may be occluded by other objects, or by part of the object surface itself. In our current implementation, the local self occlusion of a vertex-pair is taken into account by not allowing occluded viewpoints to appear as solutions in the (ϕ, ψ) look-up tables. We do not solve the full hidden line problem to determine global occlusion, although this is desirable for a full solution to the optimization problem.

Next we consider the definition of an appropriate error metric which describes the uncertainty in the (ϕ, ψ) parameters as a function of (ϕ, ψ) orientation, or equivalently, with respect to position on the viewsphere.

3 An Effective Viewpoint

3.1 Image Segmentation Error

In our current approach to segmentation [Canny], we rely on zero crossings in the second derivative of image intensity to define the location of geometric edge and vertex features. There are many phenomena that can cause the location defined by the second derivative to be in error with respect to the ideal location corresponding to the image projection of a given object feature. Some of the more significant effects:

- Complex image intensity behavior that does not correspond to the simple step edge model employed to detect object boundaries (e.g. corners and junctions).
- Boundary characterized by texture, rather than simple intensity discontinuity.
- Quantization in image intensity and spatial resolution.
- Random uncertainty in sensor intensity values (Usually a small effect.)

These effects contribute to uncertainty in the detected boundary element locations. In our approach these "edges" are then linked, and the resulting boundary chain is approximated by straight line segments [Asada and Brady]. This process introduces additional error in the segmentation geometry. Some of the major effects here are:

- Boundary chains with low curvature do not have well defined segment endpoint location.
- Image spatial quantization introduces significant uncertainty in the chain curvature measurement.

Finally, even more uncertainty is introduced by the necessary extrapolation of line segments to form vertices between endpoints of lines that should (ideally) meet. This extension is necessary because some portions of the boundary are missing due to poor edge detector performance near junctions and in the case of low contrast boundary intervals.

The cumulative result of these various phenomena in our current implementation is that edge orientation is accurate to about 5^o, and vertex location is accurate to a radius of several pixels. These errors can be much larger for the case of line segments with lengths that are comparable to the vertex uncertainty (5-10 pixels).

The error in edge orientation is the focus of our discussion. Next we consider the relationship between this orientation error and the (ϕ, ψ) parameters.

3.2 A Rotation Error Metric

The equational systems that relate (α, β) and (ϕ, ψ) are quite non-linear and present many special solution cases. In our approach, these equations are solved numerically and stored in lookup tables. It then becomes reasonable to compute a Taylor series expansion about a particular value of (ϕ, ψ) to provide a linear expression for the parameter mapping.

We assume that we have computed the functions, $\alpha(\phi, \psi)$ and $\beta(\phi, \psi)$ (see Figure 1). Then,

$$\alpha(\phi, \psi) = \alpha(\phi_0, \psi_0) + \frac{\partial \alpha}{\partial \phi}(\phi - \phi_0) + \frac{\partial \alpha}{\partial \psi}(\psi - \psi_0) \quad (1)$$

$$\beta(\phi, \psi) = \beta(\phi_0, \psi_0) + \frac{\partial \beta}{\partial \phi}(\phi - \phi_0) + \frac{\partial \beta}{\partial \psi}(\psi - \psi_0) \quad (2)$$

where the indicated derivatives can be computed numerically.

The Jacobian, J, of the parameter mapping is given by,

$$J = \begin{vmatrix} \frac{\partial \alpha}{\partial \phi} & \frac{\partial \alpha}{\partial \psi} \\ \frac{\partial \beta}{\partial \phi} & \frac{\partial \beta}{\partial \psi} \end{vmatrix} \quad (3)$$

Naturally, if J vanishes, then the mapping is not defined for that particular viewpoint [Whitney]. We can solve the expansion equations for the variation in (ϕ, ψ) as follows:

$$\Delta \phi = \frac{\Delta \alpha \frac{\partial \beta}{\partial \psi} - \Delta \beta \frac{\partial \alpha}{\partial \psi}}{J} \quad (4)$$

$$\Delta \psi = \frac{\Delta \beta \frac{\partial \alpha}{\partial \phi} - \Delta \alpha \frac{\partial \beta}{\partial \phi}}{J} \quad (5)$$

Assuming that the errors in (α,β) are independent and of zero mean, we can derive an estimate for the variance in the Euclidean distance between the (ϕ,ψ) estimate and the mean, (ϕ_0,ψ_0). We denote this squared distance by $\sigma^2_{\phi,\psi}$. A similar representation can be defined for the variance in the measured projection angles, i.e. $\sigma^2_{\alpha,\beta}$. The ratio of these variances is given by,

$$\frac{\sigma^2_{\phi\psi}}{\sigma^2_{\alpha\beta}} = \frac{\left[\left(\frac{\partial\beta}{\partial\psi}\right)^2 + \left(\frac{\partial\alpha}{\partial\psi}\right)^2 + \left(\frac{\partial\alpha}{\partial\phi}\right)^2 + \left(\frac{\partial\beta}{\partial\phi}\right)^2\right]}{J^2} \tag{6}$$

The ratio, $\frac{\sigma^2_{\phi\psi}}{\sigma^2_{\alpha\beta}}$, is a measure of the sensitivity of the computed model rotation parameters to errors in the segmentation. If this ratio is much larger than one, the variation in (ϕ,ψ) becomes too great to provide an effective filter for incorrect model assignments. Typically, it is necessary to bound a valid (ϕ,ψ) transformation cluster with a tolerance radius of approximately 5°, in order to eliminate false assignment hypotheses. Thus, a ratio of one, or less, is necessary to support the application of viewpoint consistency with respect to the rotation parameters, (ϕ,ψ).

3.3 Effective Viewing

We are now in a position to define the concept of effective viewing. An effective vertex-pair is one which provides a precise estimate of the coordinate transform between the three dimensional reference frame and the image projection reference frame.

A single vertex-pair cannot be effective over all viewpoints. There are obvious degenerate viewing conditions where the transformation,

$$(\alpha,\beta) \rightarrow (\phi,\psi)$$

is not defined. For example, when the viewing direction is collinear with the spine, the corresponding transform equations do not have a solution.

The error metric that we have just defined predicts these degeneracies as illustrated in Figure 10. In this figure, we have projected $\frac{\sigma^2_{\phi\psi}}{\sigma^2_{\alpha\beta}}$ onto the viewing sphere. In the left column, a coplanar vertex-pair is shown in the upper pane. The spine is horizontal and the displayed viewpoint is somewhat above the plane of the vertex-pair. One of the edges is quite foreshortened at this viewpoint. The viewsphere for this case is shown in the lower left pane. The eyepoint for this image of the sphere is the same as that of the vertex-pair in the upper pane. The intensity in this image is proportional to the ratio,$\frac{\sigma^2_{\phi\psi}}{\sigma^2_{\alpha\beta}}$. The error metric becomes high at the equator of the viewsphere, since this corresponds to the locus of viewpoints that lie in the plane of the vertex-pair. The edges and spine of the vertex-pair all collapse into a single line for this set of viewpoints. Singularities exist at the points the view axis is collinear with any of the edges or the spine.

There is also a region of relatively high error at the poles of the viewsphere. These viewpoints correspond to looking normal to the plane of the vertex-pair. This is reasonable, since the projected angles of edges lying in a plane, have the least variation with tip and tilt of the plane when the viewpoint is normal to the plane.

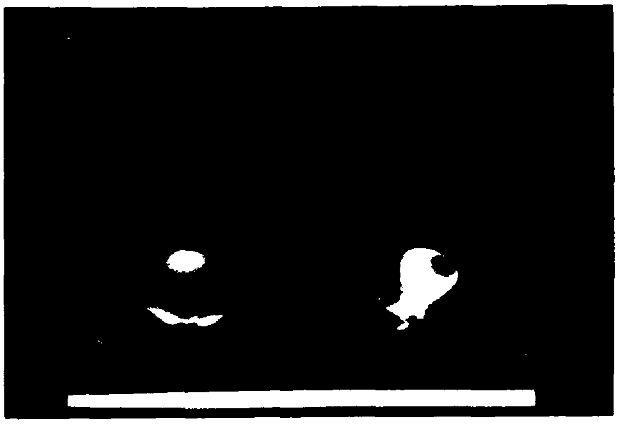


Figure 10: Two examples of the error metric projected onto the viewing sphere. The upper panes show the vertex-pairs used to generate the spheres shown in the lower pane.

The right column shows the same calculation except that the vertex-pair is not coplanar. The first edge is now inclined upw͏ard out of the horizontal plane at 45°. There are still two great circles of high value on the viewsphere. These correspond to viewpoints lying in the two planes defined by the spine and each of the edges. The error associated with normal views of each of these planes now contributes to a rather complex distribution over the sphere. The dark areas on the sphere correspond to effective viewpoints.

4 The Composition of Multiple Vertex-Pairs

4.1 Independent Segmentation Error

If we assume that the variation in the projected angles of vertex-pairs is due to statistically independent segmentation error, then a composite error metric for a set of vertex-pairs can be defined. That is,

$$< \frac{\sigma^2_{\phi\psi}}{\sigma^2_{\alpha\beta}} > = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{\sigma^2_{\phi\psi}}{\sigma^2_{\alpha\beta}}\right)_i \sigma^2_{\alpha\beta_i} \tag{7}$$

Where $\sigma^2_{\alpha\beta_i}$ is the expected variance in the angles associated with a particular vertex-pair in the segmentation. In general, the variance of edge angles in the segmentation is not directly related to the associated model vertex-pair. The accuracy of angle determination between image boundary segments is inversly proportional to edge length. The length is controlled by occlusion and edge contrast effects, and is not closely related to the projected edge length of the ideal object edge.

Therefore we can simplify the composition process by assuming that the variance $\sigma^2_{\alpha\beta_i}$ is the same for all vertex-pairs grouped from the segmentation, and further that this variance is the worst case value corresponding to the shortest acceptable segmentation edges. The resulting composite error metric is simply the average of the error metric for the constituent vertex-pairs.

4.2 An Example

Earlier this year we carried out an experiment with infra-red targeting data taken from the air attack by the U.S. on Libya. These images were digitized from a video tape and are of rather low contrast and have significant random noise effects. An example of the match of a model of the Russian IL-76 transport aircraft is shown in Figure 11. The model is shown superimposed on the image data.

We have repeated this experiment to observe how the effective viewing metric behaves for the individual vertex-pairs that were selected in the model. Figure 12 shows the error projection for a well-behaved vertex-pair. The usual degeneracy and error distribution is evident. In Figure 13 the error display indicates that this vertex-pair is almost completly useless. There is only a small portion of the viewsphere that provides an effective viewpoint. The main reason that this vertex-pair is so ineffective is that one of the edges is nearly collinear with the spine vector. In this configuration, the projected angle for that edge does not vary significantly for a broad range of rotations about the spine. Thus, the variance in (ϕ, ψ) is large for large regions of the viewsphere.

The composite error map in Figure 11 shows that the composite is still effective although the general variance level has been increased over large areas of the sphere. The composite is computed for five vertex-pairs that were manually selected from the model. This result indicates that the match robustness is probably low and that we were somewhat fortunate to get a unique match in this experiment.

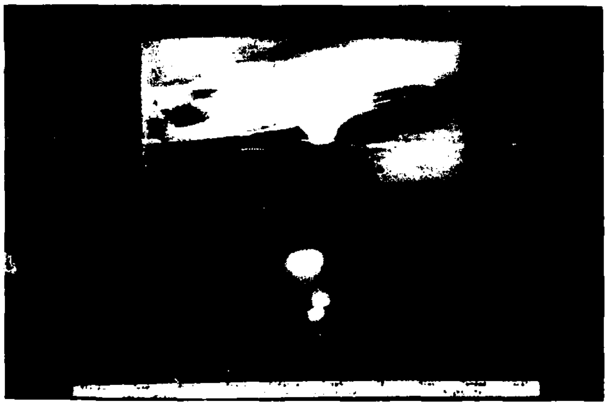


Figure 11: A match on infra-red image data. The composite error metric is shown projected on the viewing sphere.

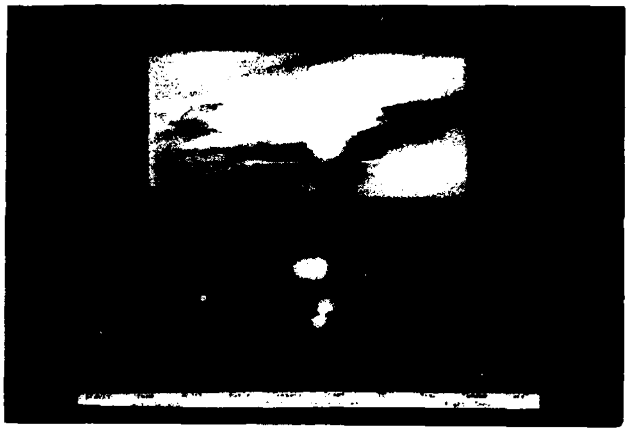


Figure 12: A well behaved vertex-pair and its associated error sphere.

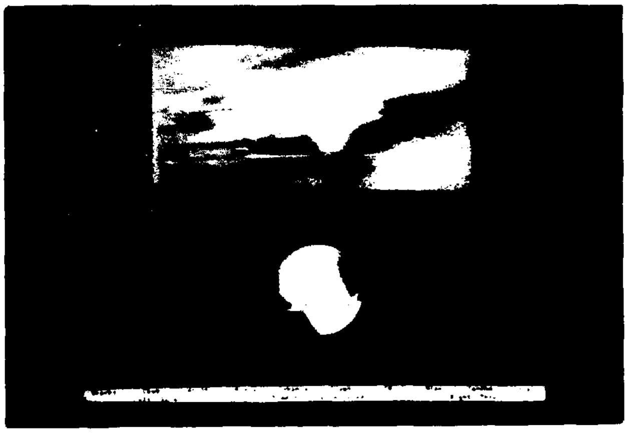


Figure 13: A poorly behaved vertex-pair and its associated error sphere, showing large areas of high error.

5 Automatic Vertex-Pair Selection

5.1 Computing the Optimal Vertex-Pair Set

A major application for the vertex-pair error metric is in automatic vertex-pair selection. Given a large number of vertices in the model, it is necessary to avoid generating a correspondingly large number of model vertex-pairs. In particular, it would be desirable to have sufficient vertex-pairs defined so as to provide some necessary number of vertex-pairs with good local error behavior for every possible viewpoint. We want to minimize the total number of model vertex pairs, M, because the complexity of the algorithm is of order $M(N-1)^2$ (sec. 1.3).

In general, to generate all possible vertex-pairs in the model requires pairing each vertex with every other vertex. The only characteristic of the vertex-pair geometry whose effect on error can currently be evaluated for a general viewpoint is the pair of angles between the spine and the edges. If either of the edges is nearly collinear with the spine, the error value will be high over a wide range of viewpoints. By limiting the incident angle at the base vertex, we can reduce the number of model vertex-pairs generated from $n^2 - n$ to $c(n^2 - n)$ where c is the fraction of $(0 - \pi)$ allowed as an incident angle and n is the number of model vertices.

The proposed next step in generating an active set of model vertex-pairs is ranking the $c(n^2 - n)$ model vertex-pairs generated above according to the surface area of the viewsphere in which they demonstrate low error. First, we apply a simple local visibility criterion to determine the area of the viewsphere within which each vertex-pair will be visible (the intersection of outside half spaces of the faces incident on the vertex-pair). Then we may compute error maps for the remaining areas of the viewsphere for each vertex-pair. The surface area of the viewsphere which is visible and of low error for a model vertex-pair is the value of that vertex pair for the purpose of active set selection.

Finally, we may select vertex-pairs by searching linearly through the ordered set of candidates until all of the points on the viewsphere are addressed by some minimal number of vertex-pairs or we run out of vertex-pairs. This will result in the optimal group of model vertex-pairs being selected. If we run out of model vertex-pairs before completely covering the viewsphere, those remaining areas of the view sphere are still covered by vertex-pairs which exhibit high error there.

5.2 Using High Error Vertex-Pairs

Certain viewing conditions require the use of high error valued vertex pairs. These would occur, for example, when looking at a model from a viewpoint such that all visible surfaces lie on a plane orthogonal to the view axis (eg. the top of a cube). In this case, only coplanar vertex-pairs would be visible, and they would be viewed from a point on the viewsphere where their error value is known to be high. However, although the only matching features available have a high error value, they may still return the correct transform values.

High error means a vertex-pair is unable to precisely determine the (ϕ, ψ) rotation at that viewpoint. It will therefore propose a number of possible solutions (ϕ, ψ) for a given (α, β). Areas matched with high error vertex-pairs will therefore be more liable to produce false positive responses.

6 Conclusion

We have defined an error metric that quantifies the matching precision of the vertex-pair match feature as a function of viewpoint and geometry. With this tool, we were able to define the concept of an efficient viewpoint, using minimal sets of demonstrably powerful vertex-pairs to accomplish the matching task.

Our next topic will be to consider the role of viewpoint based error in the clustering process used to determine the presence of matches. Currently, the precision of a model vertex-pair is not considered in transform clustering. Imprecise model vertex-pairs simply propose more solutions. Each solution is treated as a separate, equally viable result. We believe the error metric presented here may be useful in determining the likelihood that a transform cluster is a false positive match, resulting from the coincidental contributions of a number of high error valued vertex-pairs.

We will also explore a more formal understanding of the error values of vertex-pairs as a function of geometry. We hope that this may facilitate the generation of an optimal set of vertex-pairs for matching by replacing more of the ranking by error behavior with direct ranking by geometry.

References

[Thompson and Mundy] Thompson, D., Mundy, J.L., "Three-Dimensional Model Matching From an Unconstrained Viewpoint", Proc. IEEE Robotics and Automation, 1987, pp. 280.

[Canny] "Finding Edges and Lines in Images", MIT Artificial Intelligence Laboratory Report, AI-TR-720.

[Asada and Brady] Asada, H. and Brady, M. "The Curvature Primal Sketch", Proc. IEEE Workshop on Computer Vision: Representation and Control, 1984.

[Lowe] Lowe, D., "Perceptual Organization and Visual Recognition", Kluwer Academic Publishers, Boston, MA, 1985.

[Ayache and Faugeras] Ayache, N. and Faugeras,O., "A New Method for the Recognition and Positioning of 2D Objects," Proc. 7th International Joint Conference on Pattern Recognition, 1984.

[Huttenlocher and Ulmann] Huttenlocher, D.P. and Ullman, S., "Object Recognition Using Alignment," Proc. 1st International Conference on Computer Vision, 1987.

[Grimson and Lozano-Perez] Grimson,W.E.L., Lozano-Perez, T.,"Search and Sensing Strategies for Recognition and Localization of Two and Three Dimensional Objects," Proc. 3rd International Symposium on Robotics Research, MIT Press, 1985.

[Ikeuchi] Ikeuchi,K. "Precompiling a Geometrical Model Into an Iterpretation Tree for Object Recognition in Bin-Picking Tasks," Proc. DARPA Image Understanding Workshop, 1987.

[Ikeuchi and Kanade] "Modeling Sensor Detectability and Reliability in the Configuration Space for Model-Based Vision," Carnegie Mellon Report CMU-CS-87-144.

[Whitney] "On Singularities of Mappings of Euclidean Spaces. I. Mappings of the Plane Onto the Plane," Annals of Mathematics, Vol. 62, Nov., 1955.

[Chakravarty] Chakravarty, I., "The Use of Characteristic Views as a Basis for Recognition of Three Dimensional Objects.," Ph.D Thesis, Rensselaer Polytechnic Inst., Troy, N.Y.,1982.

Core Knowledge System: Storage and Retrieval of Inconsistent Information

Thomas M. Strat and Grahame B. Smith

Artificial Intelligence Center
SRI International
333 Ravenswood Avenue
Menlo Park, California 94025

Abstract

We describe an information storage and retrieval mechanism that avoids the requirement of consistency maintenance imposed by traditional knowledge-based systems. By viewing data as opinions rather than facts, the system is able to combine knowledge that is generally accepted as being true with data that may be from unreliable sources. We also give a formal account of the semantics of the approach. The information management system described has been implemented and used to store information derived from image processing along with opinions from other sources about objects in the visible world. It appears to be well suited for the requirements of an autonomous robot, and for information storage in general.

1 Introduction

Database management systems are widely used to store and retrieve facts about the world. They provide a convenient and efficient means to access vast amounts of data and have been the basis of data management in many varied applications. In *knowledge-based systems*, they have been used to manipulate data that encodes the knowledge that the system uses to carry out its activities. In these circumstances it has been usual to impose upon the data the requirement of consistency. The database is usually expected to be a consistent collection of facts presumed to be true about the world. Conflicting items of information are filtered from the database; else their presence would result in erroneous conclusions. For this reason, great care must be taken to ensure that false data are not inserted into the database, and much effort has gone into developing procedures for ensuring the integrity of the data store. However, there are numerous knowledge-based system applications, autonomous robots being one, in which the requirement of database consistency is inappropriate.

Whenever information is provided by more than one source or when the validity of that information is undeterminable, it may be impossible to satisfy the requirements of consistency. For example, suppose a friend tells us that the fishing in Lake Mohunk is great this time of year, and we wish to store this information in a database. We would first be obliged to ensure that it is not in conflict with anything already present in the database, and we would be required to ensure that that remains the case. In general, whether or not we believe a statement is valid depends on a number of factors:

Credibility of the source If our friend is an avid fisherman who just returned from the lake, we should be much more willing to accept his statement than if the source of information was hearsay.

Believability of the statement — If Lake Mohunk is the center of a fishing resort where fishing has been great in the past, we would likely accept our friend's statement readily; if we know that the lake has been subject to pollution we would not.

Consistency with other statements — We would be unwilling to believe our friend if we had polled six fishermen and all the others told us that fishing at the lake is poor.

Intended use of the information — If we were just curious about our friend's recent fishing trip, we would accept his statement, but if we were planning a two-week vacation to Lake Mohunk, we might be unwilling to accept it without additional confirmation.

The traditional knowledge-based system design makes it difficult or impossible to take all these factors into account. In this paper we introduce an information storage system that was designed to overcome this limitation.

2 Core Knowledge System

The approach to database design we describe in this paper was motivated by the need to store information about the visual world in an autonomous robot. In particular, we have developed a Core Knowledge System (CKS) [4] to serve as a central information manager in a robot. A major component of CKS is a database that encodes the spatial and semantic properties and relations in the robot's environment. Because CKS accepts data from a wide variety of sources (vision and range sensors, maps, reasoning systems, and even humans), it is natural to expect that this information will be mutually inconsistent and error-prone.

The CKS database is a storehouse of the *opinions* of many agents. It is not a database of facts. This design has some unusual properties- properties that allow it to function as a central repository of information for a community of processes. It also renders much of the previous research in database design inapplicable and has spurred us to develop new techniques for the storage and retrieval of multiple opinions.

The community-of-processes architecture adopted for the CKS requires that processes be able to communicate their opinions with one another and without undue interference from processes with competing views. The formalization and use of an opinion base in lieu of a database gives rise to the following important CKS features:

- The ability to store information that is *inconsistent*.
- The ability to *integrate* multiple opinions and to allow that integration to depend upon the intended use of the information.
- The ability to *separate* one source's opinions from another's.

These goals have been achieved while retaining an ability to incorporate general knowledge that is universally accepted as being true. An opinion base appears to us to be a better model of how humans store information than is a database of consistent facts.

3 Logical Interpretation of the CKS Database

Because of the presence of inconsistent information, first-order logic is insufficient for describing the semantics of the database. In what follows we will resort to a modal logic of belief as a means for interpreting CKS transactions. This logic is extremely expressive and could be used to specify a much richer collection of knowledge. However, we have purposely limited the scope of our logic as a concession to efficiency of implementation. We are, after all, designing the CKS as the core component of an autonomous vehicle, and implementation issues cannot be ignored. For this reason, the ultimate design is a compromise between *the ability to express complicated statements involving* beliefs of multiple agents and the ability to retrieve relevant information quickly. Throughout the design, we have been guided by the requirements of autonomous vehicles and have adopted what we feel to be an efficient implementation that does not sacrifice the ability to express the information that must be communicated among the various processes of an autonomous system.

3.1 Semantics

Information is stored in the CKS in the form of data tokens. A *data token* is a framelike object whose internal structure is related to the semantic attributes of the object. Among other things, a data token contains information about the spatial location of an object and some domain-specific properties that are believed to be true about the object. For example, a portion of a data token, `token01`, might contain:

```
token01
  :SPATIAL-DESCRIPTION  (V1 V2)
  :SEMANTIC-DESCRIPTION (LARGE RED POST)
  :HEIGHT               7.3
      . . .
```

Here the *spatial description* is a list of volume specifications (as we have previously described [4]) in which the object is presumed to lie. The *semantic description* is a list of properties that the asserting agent believes to be true about `token01`.

The precise meaning of CKS transactions is specified with the aid of modal logic. This logic *consists of the following components*:

Object Constants The collection of all data tokens and potential data tokens in the database.

Function Constants -- None.

Predicates — The set of vocabulary words and the set of possible spatial descriptions.

We adopt the usual syntax for forming well-formed formulas (WFFs) over these symbols as well as all the standard axioms of first-order logic. In addition, we assume the existence of implicit axioms that allow us to infer implicit spatial relationships and to make use of semantic relationships encoded in a semantic network:

- $(\forall x)[V_i(x) \rightarrow V_j(x)]$
  whenever the volume denoted by V_i is completely contained within the volume denoted by V_j, where V_i and V_j are spatial descriptions.
- $(\forall x)[W_i(x) \rightarrow W_j(x)]$
  whenever W_i and W_j are connected by a 'subset' relation in the semantic network.
- $(\forall x)[W_i(x) \rightarrow \neg W_j(x)]$
  whenever W_i and W_j are connected by a 'disjoint' relation in the semantic network.

We employ a modal operator $\mathbf{B}_i$ to be interpreted as meaning "agent i believes that" so that $\mathbf{B}_i[\Phi]$ is interpreted as "Process i believes Φ is true." This operator is similar to that described by Moore [3] and by Konolige [2]. However, our axiomatization is different, as we only develop here those formulas that are needed *to interpret the action of the database.*

The following axiom schema provides the modal operator with the semantics we desire:

- $\Phi \rightarrow \Psi \Rightarrow \mathbf{B}_i[\Phi] \rightarrow \mathbf{B}_i[\Psi]$ — The modal operator $\mathbf{B}_i$ is closed under deductive inference.

Significantly, this axiomatization does not require that $(\mathbf{B}_i[\Phi] \vee \mathbf{B}_i[\neg\Phi])$ is true, nor does it require that $(\mathbf{B}_i[\Phi] \wedge \mathbf{B}_i[\neg\Phi])$ is false. However, it does require $(\mathbf{B}_i[\Phi] \vee \neg\mathbf{B}_i[\Phi])$ to be true and $(\mathbf{B}_i[\Phi] \wedge \neg\mathbf{B}_i[\Phi])$ to be false. This arrangement allows conflicting beliefs to exist without corrupting the database.

For any proposition Φ, an agent may have any of four states of belief, which are listed below. An example is given of an English statement that would be encoded by each of these states of belief. In the example $\Phi \equiv House(\text{object})$, and $\forall x\ Car(x) \Longrightarrow \neg House(x)$ is included in the semantic network:

• "The object is a house."	$(\mathbf{B}_i[\Phi] \wedge \neg\mathbf{B}_i[\neg\Phi])$
• "The object is a house and it's also a car."	$(\mathbf{B}_i[\Phi] \wedge \mathbf{B}_i[\neg\Phi])$
• "The object is blue."	$(\neg\mathbf{B}_i[\Phi] \wedge \neg\mathbf{B}_i[\neg\Phi])$
• "The object is a car."	$(\neg\mathbf{B}_i[\Phi] \wedge \mathbf{B}_i[\neg\Phi])$

3.2 Insertions

An insertion is of the form

```
(INSERT-DATA-TOKEN token01)
```

where `token01` is an instance of a data token:

```
token01
  :SPATIAL-DESCRIPTION  (V1 V2)
  :SEMANTIC-DESCRIPTION (LARGE RED POST)
  :HEIGHT               7.3
```

Executing (INSERT-DATA-TOKEN token01) has the same effect as asserting

$$\mathbf{B}_i[V_1(\texttt{token01})] \wedge \mathbf{B}_i[V_2(\texttt{token01})] \wedge \\ \mathbf{B}_i[LARGE(\texttt{token01})] \wedge \mathbf{B}_i[RED(\texttt{token01})] \wedge \\ \mathbf{B}_i[POST(\texttt{token01})]$$

Assertions about objects can only be in the form of a conjunction of properties. This approach allows an agent to assert that token02 is a GREEN HOUSE but does not allow him to say, for example, that token02 is *either* a HOUSE *or* a BARN. If an agent desires to convey this information, he must rephrase it as, for example, token02 is a BUILDING, with the attendant loss of information. If it is truly important for a process to convey this disjunction precisely, the vocabulary should be extended to include HOUSE-OR-BARN as an acceptable term. It is our expectation that processes will need to communicate only in terms similar to those that have evolved for human communication, and as a result, that there will be little need to resort to such artificial constructs as HOUSE-OR-BARN.

3.3 Queries

The syntax of the query language is provided in Appendix A. The language differs from traditional query languages because the CKS database contains information that is both incomplete and inconsistent. The query language must provide the user with a means for discerning multiple opinions. For this reason, the query language qualifiers APPARENTLY and POSSIBLY are provided to make these distinctions. Loosely speaking, a WFF is true "apparently" if there is some agent that believes it. It is true "possibly" if there is some agent who believes it or there is no agent that believes that it is false. These notions will be formalized shortly.

As can be seen in Appendix A, queries are either simple or compound. Simple queries are a list of three items: a *qualifier*, a *relation*, and an *argument*. The argument is a semantic or spatial description or something else, depending upon the identity of the relation. Each relation is in reality a three-valued predicate, with "I don't know" being an acceptable value. This allows the user of the CKS to *reason* appropriately when information is either lacking or inconsistent. Currently four relations are provided:

IN — This relation is interpreted as being satisfied only by those data tokens that are contained within the volume denoted by the argument, which must be a spatial description.

IS — This relation is satisfied by those tokens that belong to the class denoted by the argument, which must be a semantic description.

HAS AS PART — This relation is satisfied by those data tokens of which the argument is believed to be a component.

IS PART OF — The data tokens that satisfy this relation are those that are believed to be components of its argument.

Additional relations may also be defined for use in retrievals—this facility is described in Section 3.4.

Each relation divides the universe of objects into three sets relative to its argument(s): those that are known to satisfy the relation, those that are known to not satisfy the relation, and those for which no determination is possible. The qualifiers "apparently" and "possibly" are used to indicate which sets are desired in any particular query. To describe the semantics of qualified queries, there are two cases to consider: these queries for which only a single agent has provided a relevant opinion, and those for which more than one agent is involved.

First, we enumerate the situations in which there is only a single agent. Φ is "apparently" true for the following combinations of beliefs of an agent.

Belief about Φ **APPARENTLY** Belief about $\neg\Phi$	$\mathbf{B}_i[\Phi]$	$\neg\mathbf{B}_i[\Phi]$
$\neg\mathbf{B}_i[\neg\Phi]$	Yes	No
$\mathbf{B}_i[\neg\Phi]$	Yes	No

Similarly, Φ is "possibly" true for the following combinations of beliefs.

Belief about Φ **POSSIBLY** Belief about $\neg\Phi$	$\mathbf{B}_i[\Phi]$	$\neg\mathbf{B}_i[\Phi]$
$\neg\mathbf{B}_i[\neg\Phi]$	Yes	Yes
$\mathbf{B}_i[\neg\Phi]$	Yes	No

Before providing exact formulas for the interpretation of CKS queries, we examine those situations in which Φ is "apparently" and "possibly" true when there are two agents involved.

Φ is "apparently" true for the following combinations of beliefs of two agents.

Beliefs of Agent 2 **APPARENTLY** Beliefs of Agent 1	$\mathbf{B}_2[\Phi] \wedge \neg\mathbf{B}_2[\neg\Phi]$	$\mathbf{B}_2[\Phi] \wedge \mathbf{B}_2[\neg\Phi]$	$\neg\mathbf{B}_2[\Phi] \wedge \neg\mathbf{B}_2[\neg\Phi]$	$\neg\mathbf{B}_2[\Phi] \wedge \mathbf{B}_2[\neg\Phi]$
$\mathbf{B}_1[\Phi] \wedge \neg\mathbf{B}_1[\neg\Phi]$	Yes	Yes	Yes	Yes
$\mathbf{B}_1[\Phi] \wedge \mathbf{B}_1[\neg\Phi]$	Yes	Yes	Yes	Yes
$\neg\mathbf{B}_1[\Phi] \wedge \neg\mathbf{B}_1[\neg\Phi]$	Yes	Yes	No	No
$\neg\mathbf{B}_1[\Phi] \wedge \mathbf{B}_1[\neg\Phi]$	Yes	Yes	No	No

Similarly, Φ is "possibly" true for the following combinations of beliefs.

Beliefs of Agent 2 POSSIBLY Beliefs of Agent 1	$\mathbf{B}_2[\Phi] \wedge \neg\mathbf{B}_2[\neg\Phi]$	$\mathbf{B}_2[\Phi] \wedge \mathbf{B}_2[\neg\Phi]$	$\neg\mathbf{B}_2[\Phi] \wedge \neg\mathbf{B}_2[\neg\Phi]$	$\neg\mathbf{B}_2[\Phi] \wedge \mathbf{B}_2[\neg\Phi]$
$\mathbf{B}_1[\Phi] \wedge \neg\mathbf{B}_1[\neg\Phi]$	Yes	Yes	Yes	Yes
$\mathbf{B}_1[\Phi] \wedge \mathbf{B}_1[\neg\Phi]$	Yes	Yes	Yes	Yes
$\neg\mathbf{B}_1[\Phi] \wedge \neg\mathbf{B}_1[\neg\Phi]$	Yes	Yes	Yes	No
$\neg\mathbf{B}_1[\Phi] \wedge \mathbf{B}_1[\neg\Phi]$	Yes	Yes	No	No

By extrapolating these results, we are in a position to describe the interpretation of each query in terms of our modal logic.

- (FIND-IDS (APPARENTLY IS W4)) returns a list of those objects in the set

  $$\{x \mid (\exists i)\mathbf{B}_i[W_4(x)]\} \quad ,$$

  where W4 denotes a vocabulary word. In English, this query corresponds roughly to "Find each data token for which some agent believes W_4 is true of it."

- (FIND-IDS (POSSIBLY IS W4)) returns a list of those objects in the set

  $$\{x \mid [(\exists i)\mathbf{B}_i[W_4(x)] \vee \neg(\exists i)\mathbf{B}_i[\neg W_4(x)]\} \quad .$$

  This query can be roughly translated as "Find each data token for which some agent believes W_4 could possibly be true." The set of objects which satisfy this query will always include those objects that satisfy the *APPARENTLY* version.

- (FIND-IDS (APPARENTLY IN V3)) returns a list of those objects in the set

  $$\{x \mid (\exists i)\mathbf{B}_i[V_3(x)]\} \quad ,$$

  where V3 is a spatial description. This query can be interpreted as "Find each data token such that some agent believes it is contained within the volume V_3."

- (FIND-IDS (POSSIBLY IN V3)) returns a list of those objects in the set

  $$\{x \mid [(\exists i)\mathbf{B}_i[V_3(x)] \vee \neg(\exists i)\mathbf{B}_i[\neg V_3(x)]\} \quad .$$

  This query can be interpreted to mean "Find each data token such that some agent believes that it could possibly be in the volume denoted by V_3."

- (FIND-IDS (AND Query1 Query2))

  $\{x \mid x \in$ (FIND-IDS Query1) $\wedge\, x \in$ (FIND-IDS Query2)$\}$.

  In English, this can be translated as "Find each data token that satisfies both Query1 and Query2." It can be thought of as the intersection of the sets of tokens that satisfy each query.

- (FIND-IDS (OR Query1 Query2))

  $\{x \mid x \in$ (FIND-IDS Query1) $\vee\, x \in$ (FIND-IDS Query2)$\}$.

  This can be translated as "Find each data token that satisfies either Query1 or Query2." It can be thought of as the union of the sets of tokens that satisfy each query.

- (FIND-IDS (AND-NOT Query1 Query2))

  $\{x \mid x \in$ (FIND-IDS Query1) $\wedge\, x \notin$ (FIND-IDS Query2)$\}$.

  This query can be interpreted to mean "Find all data tokens that satisfy Query1 but that do not also satisfy Query2." In set-theoretic terms, it is the set difference of the sets of data tokens that satisfy Query1 and Query2.

In answering negative queries, the database has no closed-world assumption [1] (i.e., it does not assume that a proposition is false if it cannot be proved true), thus it avoids issues of non-monotonicity and has no need for circumscription. A negative belief cannot be inserted in the database. However, a negative belief can be deduced using the deductive inference axiom and the general knowledge incorporated in the vocabulary. For example, a process cannot assert that token03 is not a PINE. But if it does assert that the token is an OAK, (i.e., $\mathbf{B}_i[OAK($token03$)]$), the database will infer that $\mathbf{B}_i[\neg PINE($token03$)]$.

3.4 User–Defined Relations

In addition to the predefined relations, a new relation can be defined by providing a LISP function as the relation in a query. It may take any arguments that are appropriate for it, but must return T, NIL or :MAYBE. For example, a relation WEIGHS-MORE-THAN could be defined as follows.

```
(defun WEIGHS-MORE-THAN (token ref-weight)
  (let ((weight (RETRIEVE-SLOT-OPINION token
                    :WEIGHT 'LATEST)))
    (cond ((null weight) :MAYBE)
          ((> weight ref-weight) T)
          (T NIL))))
```

Programmers who write procedural relations are cautioned to write them efficiently. While the transaction parser will attempt to optimize the query evaluation, nevertheless it will sometimes be forced to evaluate the relation on a large list of data tokens.

The qualifier is used to interpret the results of the three-valued predicate's evaluation of each data token in order to construct the list of tokens that are to be returned:

APPARENTLY — The list returned contains only those tokens for which the function evaluates to T.

POSSIBLY — The list includes all those tokens for which the function evaluates to T or :MAYBE.

For the standard relations, the above description indicates the effect of a query, but not the implementation. The actual algorithm used for the standard relations is much different in order to achieve high performance on very large databases.

3.5 Discussion

There are several restrictions upon the statements that an agent can make about the world and on the types of queries that can be posed. These restrictions were necessary to enable a practical implementation of the database.

The query language only allows a limited variety of queries. Acceptable queries are limited both by their syntax and by the vocabulary of properties. The query language is not intended to be a universal language. We have designed it so that the only queries that can be posed are those that can usually be retrieved efficiently, given our database architecture. It is important to bear in mind that the limitation of the query language is one that restricts only what questions can be answered efficiently; it does not prevent the identification of data tokens that satisfy an unusual query. When faced with a question that cannot be posed as a syntactically legal query, a user can obtain the exact retrieval by first retrieving a superset of the desired data tokens with an acceptable query, and then examining each token in that set individually for satisfaction of the intended query.

4 Slot Access

The query language described in Section 3 provides the means to insert data tokens into the CKS database and to retrieve pointers to those tokens based upon spatial and semantic criteria. In this section, the various mechanisms for gaining access to the information contained within a data token are described.

Data tokens are stored as frames consisting of a number of slots. Externally, each slot has a single value. Internally, however, a separate value is maintained for each process that offers an opinion. Conceptually, an association list that pairs the process name with its opinion is stored for each slot.

```
    SLOT-NAME:     ((P1 .  V1) (P2 .  V2) ...  )
```

When retrieving a slot's value, one specifies the combination method desired to combine all values that have been previously provided for that slot. This approach makes it possible to integrate multiple opinions in a manner that is suited to the task at hand. For example, if a robot wants to determine whether its camera will have an unobstructed view over a fence, it should use that opinion that has the greatest value for the HEIGHT slot. On the other hand, if its goal is to keep all the cows in a confined area, it should be interested in the smallest value of HEIGHT.

The following function is used to store a new opinion as the value of a slot.

```
  (INSERT-SLOT-OPINION <id> <slot> <value>
                       &optional
                       <auxiliary-data-fields> )
```

INSERT-SLOT-OPINION causes <value> to be stored as the opinion of the calling process for the <slot> of the data token denoted by <id>. If the process has already provided an opinion, it is replaced by <value>. The strength of belief and time of belief may be specified in <auxiliary-data-fields>. If present, this information is stored in the internal representation and can be retrieved with RETRIEVE-SLOT-OPINION. If <slot> is not the name of a currently known slot in <id>, a new slot is created which provides the facility for a user to extend a data token so as to include a slot of his own choosing.

Slot values are retrieved from data tokens using the following function.

```
  (RETRIEVE-SLOT-OPINION <id> <slot> <arbitrator>)
```

RETRIEVE-SLOT-OPINION returns two values. The first can be viewed as the value of <slot> for data token <id>. The particular value returned is determined by <arbitrator>, which specifies how multiple opinions are to be integrated by this invocation of RETRIEVE-SLOT-OPINION. The second value contains the auxiliary data associated with the returned opinion. The <arbitrator> can be any from the following list, or a programmer can use a special-purpose procedure by supplying that procedure as the value of <arbitrator>. Additional arbitrators may be added in the future to support a larger variety of techniques for information integration.

The functionality of each choice is as follows:

DEFAULT returns the default value stored in the semantic network, and possibly inherited from a superclass. This default can be considered as another opinion of the value of the slot on the object denoted by <id>.

LATEST returns the opinion that was most recently provided to the CKS.

MIN returns the smallest of all the opinions. MIN uses arithmetic comparison if its arguments are numeric, uses alphabetic comparison for arguments that are strings or symbols, and is undefined otherwise.

MAX returns the largest of all the opinions.

AVG returns the arithmetic average of all the opinions. It ignores any opinion that is nonnumeric.

(PROCESS <proc-name>) returns the opinion that was most recently provided by the process denoted by <proc-name>.

LIST returns a list of all opinions that have been rendered.

ALIST returns an alist of all the opinions. Each pair is of the form <proc-name> . <value>, where <proc-name> is the name of a process, and <value> is the most recent opinion provided by that process.

<procedure> is the name of a function or a lambda expression provided by the programmer. Its single argument is the result that would have been returned if ALIST had been used. <procedure> should return a value that is to be viewed as the integration of all opinions. For example, a simple implementation of an opinion preference scheme could be implemented by (RETRIEVE-SLOT-OPINION <id> 'PRIORITY <slot>), where PRIORITY is defined as

```
        (defun PRIORITY (alist)
           (or (cdr (assq Process-1 alist))
               (cdr (assq Process-2 alist))
               (cdar alist)))
```

It is also possible to retrieve an entire data token, given a token <id>.

```
  (RETRIEVE-DATA-TOKEN <id>
                       &optional <slot-process-alist>)
```

Given a data token <id>, this function returns a flavor instance whose slots are filled with values as determined by the optional argument. By default, each slot receives the most recent opinion expressed by any process (i.e., LATEST). To override the default, a pair of the form (<slot> <arbitrator>) is included on the <slot-process-alist>

5 Summary

We have de:cribed an information storage and retrieval mechanism that avoids the requirement of consistency maintenance imposed by traditional knowledge-based systems. By viewing data as opinions rather than facts, the system is able to combine knowledge that is generally accepted as being true with data that may be from unreliable sources. We have also given a formal account of the semantics of such an approach.

The design differs significantly from other data management systems by allowing information to be integrated at retrieval time rather than requiring all data to be made consistent at insertion time. While this approach requires storage of a larger volume of data than would be required by other data management systems, it has several significant advantages:

- It affords the opportunity to integrate information according to the demands of the current task.
- It allows the use of information that may not be available at insertion time.
- It eliminates the need for fusing information that is irrelevant to the ongoing task.

The information management system described has been implemented and incorporated within the Core Knowledge System. It has been used to store information derived from image processing along with opinions from other sources and from users. It appears to be well suited for the requirements of an autonomous robot, and for information storage in general.

References

[1] Barr, Avron, and Edward A. Feigenbaum, *The Handbook of Artificial Intelligence*, William Kaufmann, Inc., Los Altos, California, 1981.

[2] Konolige, Kurt, "A Deduction Model of Belief and its Logic," Artificial Intelligence Center Technical Note 326, SRI International, Menlo Park, California, August, 1984.

[3] Moore, Robert C., and Gary G. Hendrix, "Computational Models of Belief and the Semantics of Belief Sentences," Artificial Intelligence Center Technical Note 187, SRI International, Menlo Park, California, June 1979.

[4] Strat, Thomas M., and Grahame B. Smith, "The Core Knowledge System," Artificial Intelligence Center Technical Note 426, SRI International, Menlo Park, California, October 1987.

[5] Ullman, Jeffrey D., *Principles of Database Systems*, second edition, Computer Science Press, Rockville, Maryland, 1982.

A Syntax for the CKS Query Language

```
<transaction>  ::=  (FIND-IDS <query>)  |
                    (FIND-CLASSES <query>)  |
                    (INSERT-DATA-TOKEN <data-token>)  |
                    (RETRIEVE-DATA-TOKEN
                           <id> <slot-process-alist>)  |
                    (INSERT-SLOT-OPINION
                           <id> <slot> <value>)  |
                    (RETRIEVE-SLOT-OPINION
                           <id> <slot> <arbitrator>)

<query>  ::=  <simple-query>  |  <compound-query>

<simple-query> ::=  (<qualifier> IN
                           <spatial-description>)  |
                    (<qualifier> IS
                           <semantic-description>)  |
                    (<qualifier> HAS-AS-PART
                           <semantic-description>)  |
                    (<qualifier> IS-PART-OF
                           <semantic-description>)  |
                    (<qualifier> <3-valued-relation>
                                          [<args> ...] )

<compound-query> ::=   (OR <query> ... <query>)  |
                       (AND <query> ... <query>) |
                       (AND-NOT <query> <query>)


<qualifier> ::=  APPARENTLY  |  POSSIBLY


<semantic-description> ::=  (<vocabulary-word> ...
                                     <vocabulary-word>)


<arbitrator>  ::=  DEFAULT  |  LATEST  |
                   MIN  |  MAX  |  AVG  |
                   (PROCESS <process-name>)  |
                   LIST  |  ALIST  |
                   <procedure>
```

IU SOFTWARE ENVIRONMENTS

Christopher C. McConnell and Daryl T. Lawton

Advanced Decision Systems
201 San Antonio Circle, Suite 286
Mountain View, CA 94040-1289

ABSTRACT

In this paper we review the development of IU environments and their significant features. We begin with a history of machine vision environments. We then detail the basic components of general IU environments with respect to representations, programming constructs, system-specific data bases and user interfaces. We look at how these components have been realized in several different systems through examples and conclude with some possible developments in the near future.

1. INTRODUCTION

There is an enormous diversity of work in image understanding, ranging from the development of algorithms to building integrated systems; from research dealing with fundamental questions to specific applications with demanding requirements. Over the past 25 years, several different software environments have been developed to support and integrate IU activity. These environments were initially based on simple constructs for programming specific machines and maintaining reusable libraries of subroutines common to vision research and applications. They have now evolved into integrated systems of considerable computational and representational power, reflecting the range of problems researchers in computer vision deal with and incorporating much of what has been learned about machine vision. These systems are referred to as **Image Understanding Environments** or **IU Environments**. In this paper we review several of them and abstract their basic functional components in order to identify underlying principles of their construction and to indicate future developments.

A key attribute of IU environments is that they abstract the types of objects and underlying computational geometries used in machine vision into useful programming constructs. This allows programming in terms of a **virtual** architecture reflecting these representations and spatial organizations. It is possible to develop algorithms as though there are individual processors associated with each pixel or each extracted image object in diverse neighborhood topologies. Abstraction is not only useful for expressing algorithms concisely and developing them rapidly but also supports machine-independent code development and integration. Abstraction is becoming increasingly important as more vision machine architectures are built. The importance of rule-based processing over spatially indexed data bases of objects, such as junctions, curves, regions, surfaces, volumes and perceptual groups, in addition to images, has become clear in the past decade. To represent such objects, image understanding environments have been built on top of powerful object-oriented programming environments which support abstract type definition, combination and inheritance mechanisms. Programming constructs in IU environments also reflect the variety of spatial organizations developed in vision research, which include local, iterative processes, pyramid and multi-resolution grid organizations and generalized Hough transforms.

IU environments also supply the necessary tools for organizing the complexity of vision research and building vision systems. In a typical development, a researcher may work with hundreds of related images, millions of extracted image objects, code written by several different people, and different software packages ranging from numerical routine libraries to expert-system shells. Some environments now provide automatic, intelligent data base mechanisms for keeping track of processing history, results and relevant code. There are also rich user interfaces which support rapid access to these data bases and for displaying objects and processing status. Many of these are highly interactive and allow great flexibility in controlling how information is presented or how an object is displayed.

An important aspect of vision work is that it almost always involves several people. IU environments directly impact the culture of such groups of researchers and engineers by enhancing communication through a common representational framework and constructs which support code-sharing and data. They allow the rapid prototyping and testing of ideas and significantly enhance the learning of novice users.

2. HISTORY

Image processing software has evolved from machine-specific software packages used for interactive image analysis to machine-independent tools and representations for building autonomous vision systems. Several trends have been a part of this evolution. The power of hardware and systems software has increased to the point where a supercomputer of a few years ago will soon sit on someone's desk. The representations and operations used in vision work have expanded from images and local pixel operations to a larger and more abstract set of objects and computational geometries for such constructs as pyramids, curves, regions, surfaces and volumes. In some systems this has been achieved through the use of powerful object-oriented programming environments. In addition, environments have integrated techniques and representations such as rule-based inference systems, hypothesis management schemes, and blackboards. User interfaces have changed from simple image displays and inflexible batch-oriented command processors to multi-window displays that allow the interactive manipulation of graphic objects and text. Databases have been added to keep track of objects, processing history and software. Current systems are now more open and extendable. They can remain viable and take advantage of new representations, new techniques and new hardware within the same framework.

Early systems designed to support human analysis of images include PAX [Joh70], and VICAR [Cas79]. A useful overview of several early systems can be found in Preston's review article [Pre81]. These systems generally assisted human interpretation of aerial and satellite images for strategic and environmental purposes. There was a heavy emphasis on interactive noise removal and application of local image operations, such as thresholding, contrast enhancement and edge extraction, to increase the visibility of features to human operators. Nearly all consisted of a subroutine library written in FORTRAN and assembly language. The libraries included utilities for the display of and conversion between different formats, geometric operations such as scaling and rotation, image transforms such as noise removal and Fourier analysis, arithmetic operators for matrices and complex numbers, image measurement subroutines for computing histograms, and basic pattern classification routines. These libraries were usually used through an interactive command processor which consisted of a top level that allowed the names of library subroutines to be typed together with their arguments. Some of these command processors grew into a more general interactive interface to the operating system as a whole.

The development of software packages and environments to support research in building autonomous vision systems began in the late 1960s and is continuing today. An exact chronology is difficult because most environments have been through many stages of development, but there are clear innovations associated with several of them. One of the earliest was associated with the Hand-eye system developed at Stanford [IJC71. IFI68]. The UMIPS [DKPR83] system from the University of Maryland consisted of a standard library of subroutines, but it also introduced programming constructs for specifying neighborhood operations over images. The basic programming constructs were efficiently implemented and could be combined to yield new and combinable neighborhood operators. UMIPS leveraged strongly off the facilities provided by the UNIX operating system and its philosophy of building small tools that can then be combined through a command language. HIPS [LCS84] is another system that made heavy use of these same abilities in UNIX. Both UMIPS and HIPS associate processing histories with images for a primitive results database.

The VISIONS OPERATING SYSTEM [Ke83] from the University of Massachusetts at Amherst was developed in the mid 1970's and continues to support the ongoing development of the VISIONS image understanding system [HR87]. It has since been through several expansions. It was one of the early systems to integrate the numerical and symbolic parts of vision research by using FORTRAN for numerically intensive image operations, and by using LISP for symbolic processing necessary for higher level vision. It contained several programming constructs for expressing neighborhood operations in registered images and pyramids and for iterating these local operations in several different ways. Databases for maintaining the status and relationship between objects in the current processing environment and for long term results were also provided. Cooperative code development among a large number of graduate students was supported by semi-automatic mechanisms for generating help files that described routines developed by users and added to the system. In this way, a growing library of software was developed by a large set of researchers. Recently, the VISIONS OPERATING SYSTEM has been used as the basis of a commercially developed IU environment called KBVision [KBV87].

The SRI Image Understanding Testbed was developed in the early 1980s as a system of hardware and software to facilitate the integration, testing and evaluation of research by application-oriented groups using results from the DARPA IU Community. It combined the low level tools and high level modules developed by several members of this community. The IU testbed was based upon a common set of hardware (a VAX 780 running Unix with a Grinnell display device) to be located at each site. The low level tools supplied by various groups included C-based definitions for a command processor, displays, routines, images, and routines for manipulating strings, lists, vectors and matrices. The defined objects and displays amounted to an effort to implement object-oriented constructs in a conventional language by defining centralized routines that could select an appropriate routine to deal with the specific qualities of an object passed to it. For example, there were functions for opening or closing images that would work on any of the several supported image formats. This did not, however, support the clean expression of inheritance. The high level modules included programs for three dimensional model-based vision, determining stereo camera transforms, stereo correlation, generalized Hough transforms, finding lines, recursive region segmentation and relaxation. These high level modules were written in different languages and most of them did not make use of the lower level tools. The Testbed probably did not become an evolving standard because it did not offer enough support for writing new programs and representations, depended on standardized hardware, and had a limited interactive interface.

Image Calc [Qua84] was developed at SRI during the same period as the SRI Testbed and introduced several significant advances. It is based on the Symbolics LISP Machine which is an interactive, single user workstation with an exceptionally powerful environment. The entire system is written in ZetaLISP and the object-oriented programming system Flavors. The uniformity provided by using LISP at all levels of vision programming, both pixel and object, supports the development of abstract objects and the integration of different levels in vision research. The LISP development environment makes rapid, focused testing and debugging possible while supporting compilation for efficiency. Image Calc provides a database caching mechanism that allows high level procedures to share computationally expensive operations without paying the penalty of doing the same operation multiple times. The user interface provides many different ways of viewing and interacting with images, and selecting operations through the use of menus.

Powervision [RM88], developed at Advanced Decision Systems, is also based on the Symbolics LISP machine. Standard abstract was defined definitions for several types of objects beyond images including curves, regions and groups. It introduced the idea of **virtual images** which are images whose contents are defined by a functional closure [SJ84] that describes how to transform the values in another image. The image is virtual in the sense that no actual storage is required and the functional closure can be arbitrarily nested. Powervision defined browsers for interacting with databases for objects, processing histories and results. Powervision has been extended in a more recent system named View [MEB⁺87]. View's main innovations are the use of Common LISP [SJ84] and CLOS [BDG⁺87] to define a machine-independent environment with object-oriented programming and explicit support for multiple representations of the same abstract object. It associates basic programming constructs, such as access and iteration, with image understanding objects, rather than with the programming language, so that these operations can be easily extended to new representations. The notion of virtual images has been extended to a larger class of objects.

An emphasis in recent environments is to integrate image understanding environments with more general AI tools. Significant examples are the SRI Core Knowledge System [SS87], interactions between the VISIONS OPERATING SYSTEM and the UMASS Generic BlackBoard [CGM86] project like AuRA [ARH87] and the CMU local map builder [SS86]. These are blackboard based systems that are designed to integrate different vision and planning modules into a working system.

There have been several IU environments developed for specific hardware architectures. These can be very roughly grouped into commercial systems and research prototypes. Some commercial systems, e.g. those developed by Bausch & Lomb, Leitz, Joyce-Loebl and others in the 1970s, were most easily accessible to a user at the command or menu level [Pre81]. They took advantage of special purpose hardware to speed up the interactive processing of images greatly. Dedicated interface devices, such as keyboards with predefined keys for basic operations and trackballs, were also included. Later commercial systems, such as the Environmental Research Institute's CytoComputer, the PIXAR and the WARP [HWW87], included subroutine libraries and custom languages for writing new routines for the hardware. Some of these custom languages are MORPHAL [Lan81], PPL [Gud79], C3PL [Pre81] and Apply [HWW87]. Programming environments associated with commercial systems tend to be tied strongly to the underlying hardware, but some do provide an interface to conventional programming languages like C and FORTRAN. The environments associated with research machines are much more open-ended. Programming languages are a necessary level of interface to them. There is a tight coupling between these experimental vision architectures and the programming constructs in image understanding environments for vision research: the machines embody the programming constructs directly. The late 1970's also saw the development of several languages oriented toward supporting image understanding on parallel SIMD and MIMD machines. Some of these languages were PascalPL [Uhr81] and MAC [Dou81]. These languages were designed to be implemented on both sequential and parallel machines.

2.1. IU ENVIRONMENT ARCHITECTURE

Current IU software environments have very similar functional components, arranged in different topologies, and referred to by different names. Common to all of them are object types used in image understanding, such as junctions, images, histograms, and programming constructs for manipulating them. Several environments are built on top of rich programming environments which support the creation of new types of objects. The environments also include several different types of databases for keeping track of such things as object instances created during an application, permanent results, and code developed by multiple users. There are highly interactive interfaces to these different databases, for accessing objects and displaying them. As an example, figure 1 shows the architecture of the Powervision environment.

3. IMAGE UNDERSTANDING OBJECTS AND PROGRAMMING CONSTRUCTS

The semantics of several IU objects and operations, generally those dealing with spatially indexed objects and operations, can be defined abstractly and powerfully in

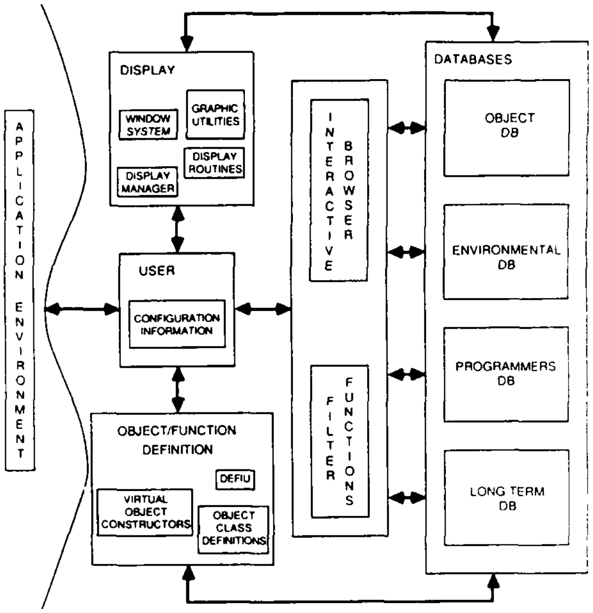


Figure 1: Powervision Environment

terms of mathematical mappings, functional composition, and relations (see work on image algebras [RW87]). Other objects, such as databases and inference processes, are more open-ended and tend to be complete modules instead *of being directly expressed in* terms of defined programming constructs. There is an important trend of building IU objects and programming constructs on top of general object-oriented programming environments [Cox84] as opposed to developing complete IU programming languages from scratch. This approach, combined with the semantics of functional mappings, enables us to begin with a very general object type corresponding to a mathematical mapping, and refining it into a class of geometric objects common in IU. These are mappings from a set of dimensional indices which can be discrete or continuous into a set of values. The power of this abstraction is that it allows object operations and combinations to be expressed with considerable generality. For example, convolution can be defined as a general operation that works on any spatially registered objects with different dimensions and representations. The ability to combine objects makes the definition of new types of objects easier.

There are four key parts to object-oriented programming: *encapsulation*, *generic functions*, late binding and inheritance. Encapsulation refers to techniques for hiding the actual implementation of representations behind an abstract functional interface. Generic functions correspond to this abstract functional interface. A generic function calls the most specific function definition associated with the class type of its arguments. The use of generic functions and encapsulation let changes be made to a representation without changing the procedures that manipulate that representation. In this way, procedures use the same interface for all objects that are of the same type even though they may have very different internal representations. Late binding means that a generic function call is bound to an actual procedure only when it is called, not when it was written by the programmer or compiled into a system. This allows a generic function call to run different procedures when it is called with different classes of objects. Since this happens at run time, the same procedure can be called on objects of *two different* representations of the same abstract type and will work correctly with both.

Inheritance is very important to organize a large object-oriented system. When new classes of objects are defined, they inherit operations from previously defined classes. Inheritance also provides a basis for the incremental development of a system while maintaining the old abilities for other representations. There are frequently a rich array of tools in an object-oriented programming environment for finding out about objects, the relationships between them and the operations defined on them. This organization and these tools significantly reduce the complexity of extending a system. Essentially, a new object is defined by saying that, "this object is just like these other objects, except for ...", without knowing all of the details of the implementations of the other objects. Code is organized into large related chunks, while allowing the ability to reuse and extend pieces without affecting the old definitions. Classes have a few general kinds of generic *functions associated with them: definitions inherited from* other classes, new definitions that are unique to a class, and definitions that **shadow** an inherited definition by modifying or replacing it. When a generic function is called on an object in a class, the new or shadow definitions take precedence over any inherited definitions.

There are three general classes of primitive generic functions associated with common geometric objects used in IU Environments: 1) Accessors, 2) Boundary checking, and 3) Iteration. More complex operations are expressed in terms of these primitive generic functions. By using the primitive generic functions, the same definition can work with objects that have very different internal representations. Definitions can still be associated with a particular representation, but this is not generally necessary since most of the processing happens in the access and iteration constructs that are already associated with particular representations. Some operations that can be defined in terms of these primitives include display routines as shown in Figure 7, and convolution as shown in Figure 5.

An **accessor** is a generic function used to get or set a property of a geometric object. Some accessors deal with attributes of an object such as non-spatial summary properties like length or average contrast for a curve. It is natural and convenient to implement at least two other general types of accessors. Given a domain index, the first type returns a spatial location, the second type returns other values associated with that location. As an example, these two accessors for a curve extracted from an image map a one dimensional domain index into the two dimensional image space that the curve lies in and into the pixel value found at that location. Geometric objects can be embedded in each other to define **neighborhoods** over which a set of values are accessed. This is useful when functions are applied over a local area with respect to some position.

A basic problem with accessors is handling indices that are outside the domain of a finite geometric object. Boundary checking operations, which compare a set of indices against the domain of an object, have several options when the indices are outside an object's domain. Options include signalling an error, returning a default value, or executing a procedure to calculate a new value. When geometric objects are combined, boundary checking can occur for the domain of each object. For example, an accessor for a region object embedded in an image can access values using its specified boundaries operations or from the domain of the image.

Another basic operation for geometric objects is to iterate an accessor over the object's domain. There are several ways to control iteration. If no particular ordering is required, the iteration can be done in parallel. To control the order of values in a sequential operation, the order that dimensions and index positions are stepped through must be specified. These correspond to the **mapping** and **walking** constructs found in VIEW.

Figures 2 through 5 show four different definitions of convolution in the UMIPS, Image Calc, Powervision and View environments. They show how basic image processing operations have been expressed in different environments. UMIPS is representative of environments that use conventional programming languages such as C and FORTRAN that lack object-oriented programming constructs for the definition of operations. Operations are defined in terms of subroutine calls, and all access and iteration is done through the constructs of the language or a basic neighborhood operator. Image Calc, Powervision and View are representative of more recent environments that are designed for powerful single user work stations. They are all written in LISP and use object-oriented programming.

Figure 2 shows an example from the UMIPS environment. In that environment, programs are defined as small pieces that are put together by the use of Unix pipes. Programs are usually defined using the programming language C. MAGIC is a facility for defining image operators over the pixels in a rectangular neighborhood. It takes care of applying the operator over an image and saving results into a registered output image. Boundary conditions are handled by insuring that neighborhoods will never reference outside of an image. **Start** sets up a neighborhood procedure by defining the size and center of the neighborhood and by setting up the input and output images. **Local** defines the actual neighborhood operation. Local is called once for each neighborhood that fits within the constraints of an image. This leaves a dead zone around the boundaries in the output image where no values are stored. The result of the operator is automatically stored at the same position in the output image.

```
/* Convolution in UMIPS Magic */
#include    "stdio.h"
#include    "cvl.h"
#define     O_NINS  1
#define     O_NOUTS 1
#include    "magic.h"

#define HEIGHT 5
#define WIDTH  5
float kernel[HEIGHT][WIDTH] = {
  {0.0025, 0.0125, 0.0200, 0.0125, 0.0025},
  {0.0125, 0.0625, 0.1000, 0.0625, 0.0125},
  {0.0200, 0.1000, 0.1600, 0.1000, 0.0200},
  {0.0125, 0.0625, 0.1000, 0.0625, 0.0125},
  {0.0025, 0.0125, 0.0200, 0.0125, 0.0025}
};

/*  Magic start up */
startup(argc, argv)
int     argc;
char    *argv[];
{
  o_height = HEIGHT;          /* Neighborhood height and width */
  o_width = WIDTH;
  o_y_cntr = o_height / 2;  /* Neighborhood center */
  o_x_cntr = o_width / 2;
  inpicf(argv[1]);
  outpicf("", 8);
}

/* Magic local operator */
local(nbd, result)
short   **nbd[O_NINS],        /* Local neighborhood */
        result[O_NOUTS];      /* Result of calculation */
{
  int       x, y;
  float     sum = 0.0;

  for (y = 0; y < o_height; y++) {
    for (x = 0; x < o_width; x++) {
      sum += nbd[0][y][x] * kernel[y][x];
    }
  }
  result[0] = sum;
}
```

Figure 2: Convolution in UMIPS MAGIC

Figure 3 shows the same algorithm programmed in Image Calc. Image Calc defines images using the object-oriented programming language Flavors [Sym86]. Generic functions associated with images include geometric and non-geometric accessors, and operations for display and file operations. **Defun-cached** defines a function that caches its results in a processing relations database as described in Section 4. Image Calc has a virtual image representation that allows low cost coordinate transformations as long as the mapping is separable into the composition of an x mapping and a y mapping. **Make-bordered-image** defines an image that transforms border image coordinates by replication or wrap around before returning the pixel found in a normal image. **With-image-elements** is an expansion environment that causes the **iref** inside of the form to be expanded into more efficient code.

Figure 4 shows the same algorithm programmed in Powervision. In this example, **defiu** corresponds to the expansion environment. It generates standard keyword arguments, error checking and recovery code, database management code, hooks for interactive function appli-

```
(defun-cached convolve-image (image kernel)
  (let* ((x-dim (image-x-dim image))
         (y-dim (image-y-dim image))
         (kernel-width (array-dimension kernel 0))
         (kernel-height (array-dimension kernel 1))
         (kernel-half-width (floor kernel-width 2))
         (kernel-half-height (floor kernel-height 2))
         (border-type :replicate) ; Border-type may be :replicate or :wrap-around
         (padded-image (make-bordered-image
                         image
                         (list kernel-half-width (- kernel-width kernel-half-width))
                         (list kernel-half-height (- kernel-height kernel-half-height))
                         :border-type border-type ))
         (into-image (make-image (list x-dim y-dim) :element-type (image-element-type image)))
         (result-fixp (image-fixp into-image))
         )

    (with-image-elements (padded-image (into-image :write))
      (loop for into-y from 0 below y-dim do
        (loop for into-x from 0 below x-dim
              for val = (loop for ky from 0 below kernel-height
                              for in-y from into-y
                              sum (loop for kx from 0 below kernel-width
                                        for in-x from into-x
                                        sum (* (aref kernel kx ky)
                                               (iref padded-image in-x in-y))))

              do (setf (iref into-image into-x into-y)
                       (if result-fixp
                           (floor val)
                           val)))))
  into-image))
```

Figure 3: Convolution in Image Calc

```
;;;  Convolution in Powervision
(defiu convolution (input kernel)
  menu utility
  input (image)
  output (smoothed)
  with (wr (array-dimension-n 1 kernel))
       (wc (array-dimension-n 2 kernel))
       (cr (// wr 2))
       (cc (// wc 2))
  in wr by wc
  boundary (img reflect)
  centerr cr centerc cc
  do (pset (loop for lr from (- cr) to cr
                 summing (loop for lc from (- cc) to cc
                               summing (* (aref kernel (+ lr cr) (+ lc cc))
                                          (pget-r img lr lc))))
          smoothed))
```

Figure 4: Convolution in Powervision

cation, boundary checking, and efficient loop generation. It contains knowledge about providing defaults and efficiently implementing common image processing idioms. The **menu** keyword indicates that this function should be put into the *utility* menu so that it can be interactively applied with the mouse. The **input** and **output** keywords define the objects that are linked together with the processing history in a database as described in Section 4. By default, the output object is a copy of the input object without the same values. The **with** keyword defines some local bindings. The **in**, **by**, **centerr** and **centerc** keywords indicate the size and center of the rectangular neighborhood being used. The **boundary** keyword indicates the type of boundary checking to be performed. This boundary checking code is generated in-line, so that the same function cannot be called on objects that require different kinds of boundary checking. The looping over each location in the input image is automatically generated around the code marked by **do**. The **pget-r**'s are access functions that get pixels relative to the current window's center. Each of these pgets automatically expands into code that reflects pixel accesses outside of the image back into the image.

Figure 5 shows the same algorithm programmed in View. This is a very general version of convolution in that both the input object and the kernel can be arbitrarily shaped, and the same routine can be called with objects that have very different representations. This generality is a result of using an object-oriented design that associates operations such as access and iteration with objects. To restrict the processing of an algorithm to some sub-region of a geometric object, a new object is created that combines a geometric object for the sub-region and the original geometric object. This is efficient since the pixels outside of the sub-region never need to be looked at.

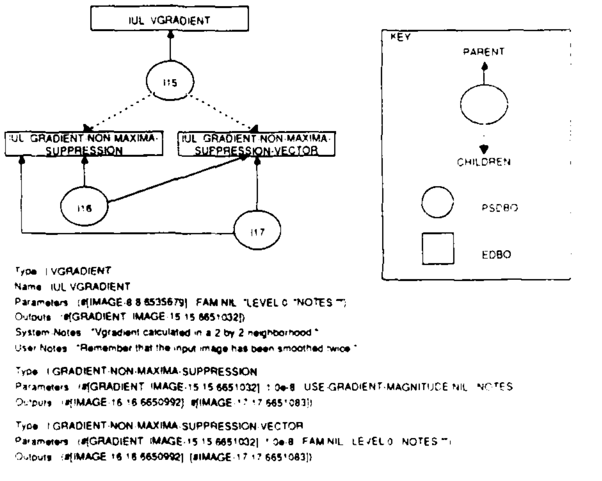
```
;;; Convolution in View
(defmethod convolution ((input geometric)
                        (kernel geometric))
  (let ((output (copy input :data nil :element-type 'float)))
    (object-access ((input :boundary 'boundary-reflect) output)
      (map-location-values
        #'(lambda (r c &rest pixels)
            (let* ((sum 0))
              (walk-values #'(lambda (value)
                               (incf sum (* value (pop pixels))))
                           kernel)
              (setf (vget output r c) sum)))
          input
          :value-accessor (neighborhood-accessor
                            (accessor input 'vget) kernel)))))
(link-to-edb 'convolution)
```

Figure 5: Convolution in View

The **defmethod** in this example indicates that this definition of convolution is a generic function associated with input and kernel objects that inherit from the geometric class. **Object-access** is an example of an expansion environment. It allows a programmer to give type and boundary checking information and to indicate that object operations that are inside the body of the form are going to be happening many times. As a result, the operations are optimized by doing the setup for the operation once when object-access is entered rather than every time the operation is done. Inside an object-access, some operations like **vget** expand into more efficient code than they would outside of the object-access. **Vget** is an accessor that given an object and a set of domain indexes returns the corresponding value in the range. The boundary checking for a vget is specified once in an object-access form rather than every time a vget is used. Here references outside of *input* are defined to reflect back into the object. **Map-location-values** is an iteration construct that calls the function marked by #' on the result of the location accessor and the neighborhood accessor for each domain index in the object. The neighborhood accessor is constructed by using the kernel to define the offsets from a base domain index and calling the accessor on each of those domain indexes. Iteration constructs that start with *map* indicate that the operations are order independent, so that they can be implemented in parallel. Those that start with *walk* are called in a specific and controllable order. The second iteration construct **walk-values** is used to walk over the values associated with each location in the kernel and to sum the product of that value and the corresponding value from the input object. The function **link-to-edb** indicates that this function should be linked with the Environmental DataBase(EDB) as described in Section 4. View separates this linkage from the definition of the operation so that a particular application can determine whether entries should be made in the EDB or not.

4. SYSTEM SUPPORTED DATABASES

There are several databases supported by IU Environments. These may be referred to as the Programmers DataBase, the Environmental DataBase, and the Long Term DataBase although they appear in several different forms in different environments. The **Programmers DataBase** (PDB) contains descriptions of processing routines and representations provided by the basic system, by library modules or developed by other users. The **Environmental DataBase** (EDB) contains objects that are produced during interactive or autonomous processing and the relationships between them. The EDB is an umbrella for two other databases, the **Object DataBase** (ODB) which contains the objects generated by an application, and the **Processing Relations DataBase** (PRDB) that records the functional relationships between the objects stored in the ODB. Figure 6 shows the EDB records generated by a call first to *vgradient*, and then to gradient-non-maxima-suppression. Gradient-non-maxima-suppression-vector, called by gradient-non-maxima-suppression, actually generated the returned result. There is usually a distinction between objects in the immediate or interactive environment and those which are stored permanently for later recall and use. The **Long Term DataBase** (LTDB) stores permanent entries from the EDB. We briefly describe some important features of these different system Databases.


IUL VGRADIENT
I15
IUL GRADIENT-NON-MAXIMA-SUPPRESSION
IUL GRADIENT-NON-MAXIMA-SUPPRESSION-VECTOR
I16
I17
KEY
PARENT
CHILDREN
PSDBO
EDBO

```
Type  I VGRADIENT
Name  IUL VGRADIENT
Parameters  (#[IMAGE-8 8 6535679]  FAM NIL  'LEVEL 0  'NOTES "")
Outputs  '#[GRADIENT  IMAGE-15 15 6651032])
System-Notes  "Vgradient calculated in a 2 by 2 neighborhood."
User Notes  "Remember that the input image has been smoothed twice."

Type  I GRADIENT-NON-MAXIMA-SUPPRESSION
Parameters  (#[GRADIENT  IMAGE-15 15 6651032]  1.0e-8  USE-GRADIENT-MAGNITUDE NIL  NOTES
Outputs  (#[IMAGE-16 16 6650992]  #[IMAGE-17 17 6651083])

Type  I GRADIENT-NON-MAXIMA-SUPPRESSION-VECTOR
Parameters  (#[GRADIENT  IMAGE-15 15 6651032]  1.0e-8  FAM NIL  LEVEL 0  NOTES "")
Outputs  (#[IMAGE 16 16 6650992]  [#IMAGE-17 17 6651083])
```

Figure 6: EDB example

4.1. OBJECT DATABASE

The Object Database (ODB) contains all of the globally accessible structures generated by an application, e.g. images, edges, regions, histograms, match results, hypothesized relations between objects and databases. With this database and interactive tools for exploring it, such as browsers, a user or process can refer to objects by their characteristics as well as their names. In this way, it provides a means of communication between different modules which can refer to the characteristics of the objects that they are interested in independently of the module that generated the object.

The ODB is supported by constructs for grouping related objects into a local database. In this way, the ODB can be organized hierarchically, which is necessary because of its potentially large size. For example, at the top level might be found images, local databases for edges and regions extracted from the same image, and significant perceptual groups created by combining primitive structures found in an image. Attributes typically associated with ODB entries are the time of creation, the object stored, parent entries, children entries, and a name or number for reference. The parents slot and the children slot are used to make connections to the entries in the PRDB. Through these connections, the processing history of an object is available.

4.2. PROCESSING RELATIONS DATABASE

The Processing Relations Database (PRDB) records function calls, their parameters and returned results. PRDB instances typically consist of a time, the function name, the parameters supplied to the function, the ODB objects used as parameters, the ODB objects returned by the function, notes added by the function writer, notes added by the function caller, and information regarding the context in which the function was applied. Instances in the PRDB can also correspond to running processes and provide a way to monitor their status.

There are often storage management operations associated with entries in the ODB and PRDB. A user or application should have control of which function calls are stored in the PRDB, so that only function calls that generate important results are present. An object can be **faded** so that the storage associated with the object can be reclaimed. When an object is faded, its ODB entry and its parent PRDB entries remain. A faded object can still be regenerated from its processing history if it is needed again. A fade operation trades storage space for processing time. When an object is saved, its ODB entry together with its processing history is saved with all of the ODB entries faded. An object can also be **forgotten** which is like fading except that as soon as all of the result ODB entries associated with a PRDB entry are faded, the ODB entries and the PRDB entry are permanently removed.

An important use of the PRDB is that function results can be cached. If a function is called with exactly the same parameters as before, the same results can be returned without calling the function again. Caching makes it easier to reuse high level building blocks that might have several operations in common at a lower level. For example, a function that goes from an intensity image to edges, and another function that goes from an intensity image to regions might both compute the gradient of the image. If the gradient operation is stored in the PRDB, then a quick check of the PRDB will show that the same operation was called before, and the old result can be returned.

4.3. LONG TERM DATABASE

The Long Term Database (LTDB) contains descriptions of all permanent ODB objects. The LTDB helps when developing an application to manage the large number of objects that are stored. Having a large number of images and preprocessed results available saves time in testing image understanding routines and makes it possible to test a routine on a variety of data sets. Each entry has a name, type, creation-date, source, documentation, directory, files, keywords and history slot. An important attribute of the LTDB is that it can store database instances as objects. Thus EDB instances can be saved as objects to retain a complete processing history so processing environments can be saved between sessions.

4.4. PROGRAMMERS DATABASE

The Programmers Database (PDB) automates the management of function and object definitions. It enables a user to find code generated by others to avoid duplicating work. This is particularly important where the environment is being used by many people working on different applications who are not directly communicating with each other. Automatic maintenance of the PDB is important because users might not spend the effort needed to maintain it manually. Entries in the PDB are automatically updated each time a file is compiled. Entries in the PDB have slots for the type, name, parameters, documentation, file, author, and date of last modification.

5. USER INTERFACE

The user interface is the contact point between a user and the environment. Current interfaces to IU Environments include underlying software development tools such as editors, window-based display systems, and browsers for searching large textual and pictorial databases. The interface provides ways to view objects as text, networks, graphs, images, curves, regions, surfaces and volumes in general and natural ways. It is important that an interface be highly interactive because it is used constantly. A bad one becomes a burden very quickly. The quality of an interface can also change as a user changes. Helpful menu-based systems become tedious and wasteful as novices become adept. It is useful if the interface maintains a user model so each user can specialize their interface and individualize it through default settings.

5.1. WINDOWS

Window-based [Tei79] interfaces are powerful, simple to use, and now very common. Most systems are written using machine-specific window packages that provide basic abilities for defining information spaces and manipulating windows with respect to them. The particular displays in image understanding environments are then built on top of this basic substrate. Software standards for machine-portable window systems such as X [SG86] and NeWS [Sun87] are beginning to become available and will lead to machine-independent interfaces for IU Environments.

IU and Graphic Environments extend window systems in several ways. One of these is to use windows to distinguish between the attributes of an object and the parameters describing how an object is displayed. This allows operations such as panning or zooming to be associated with a window and then inherited by any object displayed in that window. This provides a default display context for subsequent object displays in a window. IU environments often associate a local database with windows to maintain the history of displays at the window. This is useful for accessing a previous display without going through the original procedures that generated it and for displaying time-varying information such as image sequences. Windows can also be **linked** so that changing the information in one window causes the information in the linked window to change. This is especially useful for multi-resolution views of objects. Windows with different display types can also be linked. A typical example is a text browser linked to an object display window. When different objects are found using the browser, they are automatically displayed in the object display window.

5.2. DISPLAY TYPES

Environments typically support several predefined types of displays for spatial objects such as images, curves, regions, vectors, and surfaces in addition to objects such as graphs, network and text. The display of spatial objects requires efficiency with respect to the underlying representation used for the object. Some representations are expressed as discrete sets of connected points while others are represented by analytically expressions. This can be done by expressing display algorithms in terms of the basic iteration functions described in Section 3 that are associated with each underlying representation. Because all objects can be accessed the same way, a small set of display functions can be used on a wide variety of objects. Figure 7 shows a simple definition of the generic function **display-object** for displaying geometric objects that lie in a two dimensional space in a two dimensional spatial display window. The display window takes care of the issues of zooming, coordinate transformations and color. The generic function only has to tell the window the location of the pixels in image coordinates. **map-locations** calls the function marked by #' with the location of each position along the two dimensional object, potentially in parallel. Thus definition will work with any point, curve or region that lies in a two dimensional space whatever its internal representation. When called on a specific object, it will run as efficiently because the generic function map-locations is defined for that object's representation.

It is important that display operations can be used flexibly to highlight the characteristics of a given object. Simple examples of this are mapping pixel values onto the range of available display intensities for optimal contrast when displaying an image or color coding objects by the values of their attributes. Instead of defining several specialized displays, this can be done using virtual objects.

```
(defmethod display-object ((object 2d-geometric)
                           (window 2d-spatial-window))
  (map-locations #'(lambda (r c)
                     (plot-pixel window r c))
                 object))
```

Figure 7: Object display method

A virtual object is a functional closure [SJ84] which can be applied to an object before it is displayed. The functional closure can be reused and combined with other functional closures to produce very complicated transformations. The same effect could be achieved with functions that generated successive objects with transformed values, but it would require creating potentially large intermediate objects for each display.

An example of the use of virtual objects for display in Powervision is shown in Figure 8. The middle window shows a contour map of an elevation image with the mouse located at 84, 24 on the terrain grid. The leftmost win.low shows a perspective surface display from that position. The rightmost window shows the same perspective view, but this time the object being displayed is a virtual object that flattens any elevation pixels that are near the original position. A function to do this flattening is shown in Figure 9. As each pixel in the terrain grid is accessed, it is tested to see whether it is near the original position and if it is, then a constant value is returned. The advantages of using a virtual object are that no large image needs to be allocated, and that the function that defines the virtual object is only called on the pixels that are actually accessed.

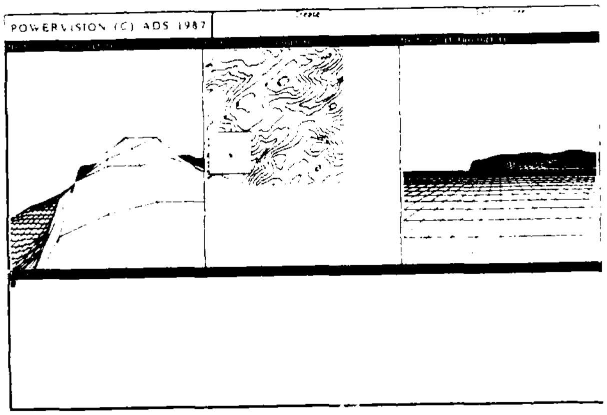


Figure 8: Virtual Object Display

```
(d :surface '((lambda (pixel)
                (if (and (< (abs (- 84 *row*)) 10)
                         (< (abs (- 24 *col*)) 10))
                    275
                  pixel))
              *image*))))
```

Figure 9: Surface Display of a Virtual Object

It is useful for graphics to be associated with an environment. These can range from a routine library for drawing points, lines, and simple geometric figures to complete rendering systems for perspective displays of surfaces. An important addition for graphics commands in an IU Environment, is that the routines have the capability of directly generating objects from the IU environment, such as images, curves and junctions in addition to a display image. This is useful for interactive segmentation and the generation of idealized data for testing algorithms.

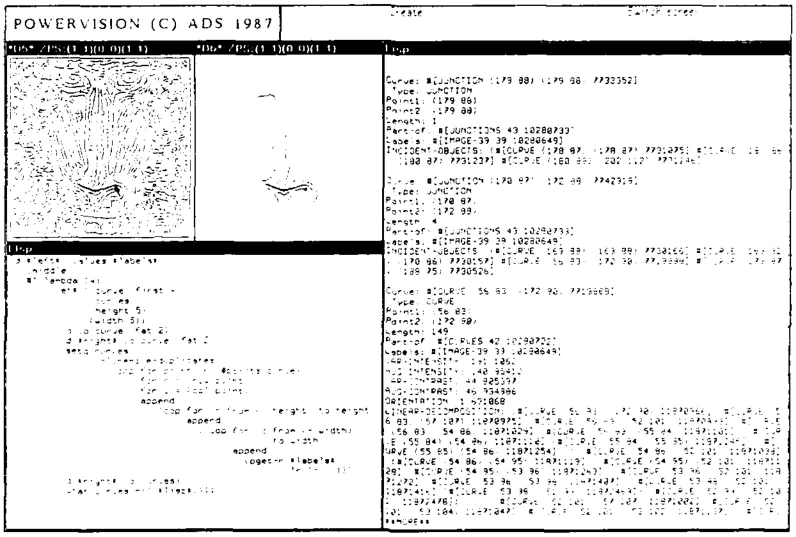
5.3. BROWSERS

Browsers are an interactive query mechanism for finding and exploring relationships between objects in databases. Among the different types of browsers that are used are text-oriented browsers, image browsers and graph browsers. All of these tend to maximize the amount of information breadth or depth presented at one time and provide a means to apply functions interactively to selected objects. Browsers can be sensitive to the type of database being queried to effectively map specialized query and link-following operations onto keys.

The text-oriented browser usually appears as a spread-sheet-like presentation of text which describes objects or functions. The text is sensitive to a pointing device, such as a mouse, and selected text is high-lighted. The display of entities is controllable in terms of the database of objects being browsed and which attributes are displayed. There are two general display modes associated with the text-oriented browser. The "object/line mode" presents each object on a single line of text divided into fields and maximizes the breadth of information that can be seen at one time. The "field/line" mode displays one object at a time with one field per line, maximizing the depth of information displayed about that one object. The object/line display is useful when looking for a specific set of objects. As the set of objects is narrowed down, the set of objects that meet selected specifications are shown. The field/line display is useful when stepping through a subset of the objects in a database, looking at each object in detail. When interacting with a database, all the elements in the text-display can not be shown and the operations is to select and order the set of objects.

Image browsers use reduced resolution pictures of objects which are sensitive to a pointing device. Usually a text browser or a filter is used to narrow down the number of images to browse. The image browser can be used to scroll through the pictures, look at their entries in the LTDB and load the objects. The pictures are generated automatically on a request to browse a specific object and then stored so that if that object is browsed again, the browser can use the previously generated picture.

Graph browsers displays objects and the relationships between them as trees or semantic nets. In the same way that the image browser shows what a stored object looks like more clearly than its textual description the graph browser shows the connections between objects more clearly. It can be used to look at the relationships between objects in the ODB and PRDB, or to trace the processing flow through object models.



```
(d *left* values *labels*
   :middle
   #'(lambda (x)
        (let* ((curve (first x))
               (curves [illegible]
               (height 5)
               (width 5))
          [illegible] curve (fat 2)
          [illegible] curve (fat 2)
          (setq curves
                [illegible]
                   (loop for [illegible] #points curve)
                      [illegible]
                      [illegible] points
                      append
                        (loop for [illegible] from [illegible] height) to height
                           append
                             (loop for [illegible] from (- width)
                                                to width
                                 append
                                   (together *labels*
                                             [illegible]
          [illegible]
          [illegible]
```

```
Curve: #[JUNCTION (179 88) (179 88) 7733352]
 Type: JUNCTION
Point1: (179 88)
Point2: (179 88)
Length: 1
Part-of: #[JUNCTIONS 43 10280733]
Labels: #[IMAGE-39 39 10280649]
INCIDENT-OBJECTS: (#[CURVE (178 87) (178 87) 7731275] #[CURVE [illegible]
  (180 87) 7731237] #[CURVE (180 89) (202 112) 7731246]

Curve: #[JUNCTION (170 87) (172 89) 7742919]
 Type: JUNCTION
Point1: (170 87)
Point2: (172 89)
Length: 4
Part-of: #[JUNCTIONS 43 10280733]
Labels: #[IMAGE-39 39 10280649]
INCIDENT-OBJECTS: (#[CURVE (163 88) (163 88) 7730166] #[CURVE [illegible]
  (170 86) 7730157] #[CURVE (56 83) (172 90) 7719888] #[CURVE [illegible]
  (189 75) 7730526]

Curve: #[CURVE (56 83) (172 90) 7719888]
 Type: CURVE
Point1: (56 83)
Point2: (172 90)
Length: 149
Part-of: #[CURVES 42 10280722]
Labels: #[IMAGE-39 39 10280649]
VAR-INTENSITY: 191 1862
AVG-INTENSITY: 140 95412
VAR-CONTRAST: 44 825397
AVG-CONTRAST: 46 734986
ORIENTATION: 1 671068
LINEAR-DECOMPOSITION: (#[CURVE (56 83) (172 90) 11870961] #[CURVE (5
6 83) (57 107) 11870975] #[CURVE [illegible] (52 101) [illegible] #[CURVE
 (56 83) (54 86) 11871029] #[CURVE [illegible] (55 84) 11871101] #[CUR
VE (55 84) (54 86) 11871118] #[CURVE (55 84) (55 85) 11871246] #[C
URVE (55 85) (54 86) 11871254] #[CURVE (54 86) (52 101) 11871039]
 (#[CURVE (54 86) (54 95) 11871119] #[CURVE (54 95) (52 101) 118711
28] #[CURVE (54 95) (53 96) 11871263] #[CURVE (53 96) (52 101) 118
71272] #[CURVE (53 96) (53 99) 11871407] #[CURVE (53 99) (52 101)
11871416] #[CURVE (53 99) [illegible] 11871463] #[CURVE [illegible] (52 10
1) 11871478] #[CURVE (52 101) (57 107) 11871202] #[CURVE [illegible]
101) (53 104) 11871847] #[CURVE (52 101) [illegible] 1187[illegible] #[CURVE
**MORE**
```

Figure 10: Display Values

5.4. INTERACTIVE DEVICES

An important attribute of any interface is the extent to which it maps context-sensitive functionality onto input devices such as terminal keys, joysticks, trackballs, and mouse keys. For example, a common operation in an IU environment is to find something that was done, apply a function to it, and then display it. Using interactive aids, this can involve establishing a text-based browser linked to the Object Database with a few clicks and mouse motions. A relevant image produced sometime during a long interactive session can be found with a few more mouse actions over the window. The selected image can then be displayed or have some routine applied to it.

Building interactive interfaces remains an art, though certain basic principles are clear and intuitively appealing [Mor79]. The interface should have explicit access to the context in which an action is occurring so it can interpret incompletely specified user actions. This generally implies a history-state mechanism. Another is that the common spatial and behavioral intuitions that people have are maintained with respect to the interactive devices. Left and right, up and down should be consistently maintaned in the object, the display and the corresponding device. When mapping object parameters onto generic control keys, if the parameters have a spatial meaning, this should be reflected in the position of keys. Excellent examples of this can be found in the surface-perspective interface and mouse key definitions used in Image Calc.

A basic interactive operation supported by many environments is a general point-and-apply function for indicating some object or object-token and applying a function to it. The applied function can be a display action or a more complicated recursive use of the point apply function itself. Figure 10 shows an example from Powervision of such a point-and-apply function called display values. In this example the display-values function is shown in the lower left window. Display-values is being applied to the *labels* image which consists, not of numbers, but of pointers to all the objects which occupy a given location in that image. This is possible because images are described as abstract types which can combine with others freely. Window *d5* shows some curves extracted from the ever-present mandrill image. When the mouse was clicked in *d5*, the function defined in :middle is executed. This function highlights the curve selected in *d5*, finds curves that are within 10 pixels of the selected curve and then displays the curves in *d6*. A textual description of some of the curves is shown in the rightmost Lisp window.

6. FUTURE DIRECTIONS

There are several ongoing trends in hardware, software and vision research affecting the future development of image understanding environments. All these factors will make IU Environments less expensive, more powerful and more common. The main driving force is the continued advance of hardware: every year, machines have more storage and are faster, cheaper and easier to connect together. Optical devices are making the storage and transmission of large amounts of digital imagery feasible. Two complementary trends in vision hardware development are worth noting. There are currently several advanced, experimental architectures being built specifically for machine vision, e.g. the Warp [HWW87], IUA [WL87] and the Connection Machine [Hil85]. Software interfaces to these machines are being built and will eventually incorporate IU environments. The other hardware trend is the development of inexpensive and modular pieces of vision or vision-compatible hardware for digitization, sensors, and pixel-level operations. In the dawning era of the personal supercomputer workstation, we will see integrated turnkey vision systems built out of these components for both industry and for mass market video workstations. We are starting to see a minor growth industry in migrating AI and vision software to SUNS and MAC IIs.

There also will continue to be advances in general software technology related to the development of Computer Assisted Software Engineering (CASE) tools that automate several tasks for developing and maintaining programs. CASE tools can keep track of code and comments through databases and browsers and allow the automatic generation of some of the code from a specification of what needs to be done. Advances in compilers allow the creation of new programming languages like Ada and object oriented programming systems [Mey87]. User interfaces will incorporate these general developments in supporting programming in workstation environments with several new types of interactive devices [Sci87]. User databases will incorporate advances in hypertext [Con87] technology for increased flexibility.

Currently many environments are being used as the basis of intelligent interactive systems for image analysis tasks in aerial photo and medical imaging interpretation. In these, much of the high-level hypothesis management and inference associated with an autonomous vision system is done by a human who can interactively match a model to an image with a small number of interactive operations. The user interface provides for the automatic storage and cross referencing of interpretations. Examples are extensions to Image Calc for terrain Interpretation, Powervision for biomedical applications, and the KBVision system for various applications. IU environments will be integrated with CAD/CAM systems to provide tools for developing workpiece inspection routines as part of the manufacturing design process.

One of the major difficulties with the library approach to IU software, in addition to problems with extendibility, was that the libraries themselves were too limited. Though vision is too complex a problem to have a uniform solution, there is a much larger body of techniques than was available before. This will be reflected in modular vision software for complex operations. Certain portions of vision research can be packaged or standardized now: constrained vision systems, stereo, iconic to symbolic mapping, surface reconstruction from sparse samples, and camera modeling software.

ACKNOWLEDGMENTS

This document was prepared by Advanced Decision Systems (ADS) of Mountain View, California, under U.S. Government contract number DACA76-85-C-0005 for the U.S. Army Engineer Topographic Laboratories (ETL), Fort Belvoir, Virginia, and the Defense Advanced Research Projects Agency (DARPA), Arlington, Virginia. The authors wish to thank Dr. Thomas O. Binford for his suggestions and Sachi Toyofuku for providing administration, coordination, and document preparation support.

REFERENCES

[ARH87] R. C. Arkin, E. M. Riseman, and A. R. Hanson. Aura: an architecture for vision-based robot navigation. In *Proceedings: Image Understanding Workshop VolumeII*, pages 417–431, February 1987.

[BDG*87] D. G. Bobrow, L. G. DeMichiel, R. P. Gabriel, S. Keene, G. Kiczales, and D. A. Moon. *Common Lisp Object System Specification.* 1987.

[Cas79] K. R. Castleman. *Digital Image Processing.* Prentice-Hall, Englewood Cliffs, 1979.

[CGM86] D. D. Corkill, K. Q. Gallagher, and K. E. Murray. Gbb: a generic blackboard development system. In *Proceedings of the AAAI-86 Fifth National Conference on Artificial Intelligence*, pages 1008–1014, Morgan Kaufmann Publishers, Inc., Los Altos, California, August 1986.

[Con87] J. Conklin. Hypertext: an introduction and survey. *Computer*, 20(9):17–41, September 1987.

[Cox84] B. J. Cox. *Message/Object Programming: An Evolutionary Change in Programming Technology.* Volume I, IEEE Software, January 1984.

[DKPR83] D. Donaldson, R. Kirby, J. Pallas, and A. Rosenfeld. *UMIPS: Univers''y of Maryland Image Processing Software.* Computer Vision Laboratory, Computer Science Center, Maryland, December 1983.

[Dou81] R. J. Douglass. *MAC: A Programming Language for Asynchronous Image Processing*, pages 41–52. Academic Press, New York, 1981.

[Gud79] B. Gudmundsson. *An Interactive High-Level Lanugage System for Picture Processing.* Dept. of Electrical Engineering, University of Linkoeping, Linkoeping, 1979.

[Hil85] W. D. Hillis. *The Connection Machine.* The MIT Press, Cambridge, Massachusetts, 1985.

[HR87] A. Hanson and E. Riseman. The visions image understanding system - 1986. In Chris Brown, editor, *Advances in Computer Vision*, Erlbaum Press, 1987.

[HWW87] L. G. C. Hamey, J. A. Webb, and I.C. Wu. *Low-Level Vision on Warp and the Apply Programming Model.* Technical Report CMU-RI-TR-87-17, Carnegie Mellon University, The Robotics Institute, July 1987.

[IFI68] IFIPS. *Stanford Handeye System.* 1968.

[IJC71] IJCAI. *Stanford Handeye System.* 1971.

[Joh70] E. G. Johnston. *The PAX II Picture Processing System*, pages 427–512. Academic Press, New York, 1970.

[KBV87] *KBVision Programmer's Manual.* Amerinex Artificial Intelligence, Amherst, Massachusetts, 1987.

[Ke83] R. Kohler and et.al. *VISIONS: Operating System Reference Manual.* Computer and Information Science Dept., University of Massachusetts at Amherst, 1983.

[Lan81] C. Lantuejoul. *An Image Analyzer.* Academic Press, London, 1981.

[LCS84] M. S. Landy, Y. Cohen, and G. Sperling. Hips: a unix-based image processing system. *Computer Vision, Graphics, and Image Processing*, 25:331–347, 1984.

[MEB*87] C. McConnell, D. Edelson, M. Black, J. Dye, and T. Esselman. *View: Programmer's Manual.* Advanced Decision Systems, Mountain View, California, October 1987.

[Mey87] N. Meyrowitz, editor. *Object-Oriented Programming Systems, Languages and Applications (OOPSLA) '87 Conference Proceedings*, Association for Computing Machinery Press, Orlando, Florida, October 1987.

[Mor79] A. Morse. Some principles for the effective display of data. *Computer Graphics*, 13:94–101, August 1979.

[Pre81] K. Jr. Preston. *Image Processing Software: A Survey*, pages 123–148. Volume 1, North-Holland Publishing Company, 1981.

[Qua84] L. Quam. *The Image Calc Vision System.* Stanford Research Institute, Menlo Park, California, 1984.

[RM88] K. Riley and C. McConnell. *Powervision.* Advanced Decision Systems, Mountain View, California, March 1988.

[RW87] G. X. Ritter and J. N. Wilson. Image algebra in a nutshell. In *Proceedings First International Conference on Computer Vision*, pages 641 645, Computer Society Press, June 1987.

[Sci87] Issue on interfaces. *Scientific American*, 257(4), October 1987.

[SG86] R. Scheifler and J. Gettys. The x window systems. *ACM Transactions on Graphics*, 5(2):79 109, April 1986.

[SJ84] G. L. Steele Jr. *Common LISP: The Language.* 1984.

[SS86] S. Stentz and S. Shafer. *Module Programmer's Guide to Local Map Builder for ALVan.* CMU Computer Science Department, June 1986.

[SS87] T. M. Strat and G. B. Smith. *The Core Knowledge System.* SRI Artificial Intelligence Center, Menlo Park, California, May 1987.

[Sun87] *NeWS Reference Manual.* Sun Microsystems, March 1987.

[Sym86] Inc. Symbolics. *Symbolics Common Lisp.* August 1986.

[Tei79] W. Teitelman. A display oriented programmer's assistant. *Int. J. Man-Machine Studies*, 11(2):157–187, March 1979.

[Uhr81] L. Uhr. *A Language for Parallel Processing of Arrays, embedded in PASCAL*, pages 53–88. Academic Press, New York, 1981.

[WL87] C. C. Weems and S. P. Levitan. The image understanding architecture. In *Proceedings of the Image Understanding Wrokshop*, pages 483–496, Los Angeles, California, February 1987.

USING PROBABILISTIC DOMAIN KNOWLEDGE TO REDUCE THE EXPECTED COMPUTATIONAL COST OF MATCHING

Azriel Rosenfeld
Avraham Margalit
Rameshkumar Sitaraman

Computer Vision Laboratory
Center for Automation Research
University of Maryland
College Park, MD 20742-3411

ABSTRACT

Matching of data structures such as digital images or labeled graphs is computationally expensive, because it requires node-by-node comparisons of the labels. If we have probabilistic models for the classes of data structures being matched, we can reduce the expected computational cost of matching by comparing the nodes in an appropriate order. This paper proves some general results about this approach, and also presents experimental results for digital images, when we know the probability densities of their gray levels, or more generally, the probability densities of arrays of local property values derived from the images.

1. INTRODUCTION

Finding matches between data structures is a basic operation in many branches of computer science. In this paper we deal with the important class of situations in which the nodes of the data structures have labels that are elements of a metric space—for example, real numbers, or vectors having real components. In such situations we can define matches quantitatively, in terms of the distances between the labels of corresponding nodes. This type of quantitative matching is widely used in fields such as image processing, computer vision, and pattern recognition, where the data structures might be images (arrays of numerical- or vector-valued pixels) or numerically labeled graphs (where the nodes might, e.g., represent segments of an image and their labels might be tuples of property values computed for these segments). We will formally define the quantitative matching problem in Section 2.

Finding "close" matches between two data structures is an expensive process, because one needs to consider many possible correspondences between the nodes of the two structures, and for each correspondence, to compute the distances between the labels of corresponding pairs of nodes, node by node. Some work has been done on reducing the amount of computation involved in this process by taking advantage of the overlaps between correspondences [1]; but we will not pursue this approach here. Rather, we will consider the possibility of reducing the expected amount of computation for a given correspondence, by computing the labels of corresponding pair of nodes in an appropriated chosen order. We will show how we can define such an order when we are given probabilistic models for the classes of data structures being matched.

In Section 3 we prove some general results about how to order the comparisons so as to minimize the expected computational cost. In Sections 4 and 5 we give some experimental results for the case where the data structures being matched are digital images.

2. QUANTITATIVE MATCHING

We will assume from now on that the data structures being matched are labeled graphs; note that digital images can be regarded as the special case in which the pixels are the nodes of the graph, the arcs connect pixels to their neighbors, and the labels are the pixel values. As already indicated, we assume that the labels are elements of a metric space.

A correspondence between the labeled graphs G and H is a graph isomorphism f of G into H. Let the nodes of G be $P_1, \ldots, P_n$, let the label of node P be denoted by $L(P)$, and let the distance between two labels L and M be denoted by $d(L,M)$. Then any correspondence f gives rise to a *distance vector* $\Delta_f \equiv \big(d\big(L(P_1), L(f(P_1))\big), \ldots, d\big(L(P_n), L(f(P_n))\big)\big)$.

In general, the *degree of mismatch* associated with the correspondence f can be defines as $||\Delta_f||$, where $||\cdot||$ is a real-valued norm. A *t-match* between G and H is a correspondence f such that $||\Delta_f|| \leq t$. Our general quantitative matching problem is to find all t-matches between G and H, for a given value of t.

We shall assume here that $||\cdot||$ is "monotonic" in the sense that, for any vector $\Delta \equiv (x_1, \ldots, x_m)$ of nonnega-

The support of the Defense Advanced Research Projects Agency and the Center for Night Vision and Electro-Optics under Contract DAAB07-86-K-F073 is gratefully acknowledged, as is the help of Sandy German in preparing this paper

tive real numbers we have $||(x_1, \ldots, x_i)|| \leq ||(x_1, \ldots, x_m)||$ for all $1 \leq i \leq m$. [This is true for many standard norms, e.g., for $||(x_1, \ldots, x_m)|| \equiv \sup_{1 \leq i \leq m} x_i, \sum_{i=1}^{m} x_i, \sqrt{\sum_{i=1}^{m} x_i^2}$, and so on.] This assumption allows us to reduce the computational cost of finding all the t-matches. In fact, for a given correspondence, if we have already computed $d\big(L(P_1), L\big(f(P_i)\big)\big)$ for a subset of the i's, and find that the norm of this tuple of d's already exceeds t, we need not compute the remaining d's, since $||\Delta_f||$ will surely exceed t. For concreteness, we shall assume from now on that $||\Delta|| \equiv ||(x_1, \ldots, x_m)|| = \sum_{i=1}^{m} x_i$.

If we have a probabilistic model for the d's, we can further reduce the expected computational cost of finding all the t-matches by computing the d's in an appropriate order. For example, suppose that for each L we know the expected value of $d(L, L')$ (averaged over the set of all L'). If we compute the d's in order of their expected size (i.e., first compute $d\big(L(P_i), L\big(f(P_i)\big)\big)$ for those $L(P_i)$'s for which the expected value is greatest, then for those for which it is next greatest, and so on), then we can expect that the value of $\Sigma d\big(L(P_i), L\big(f(P_i)\big)\big)$ will increase rapidly, so that it should only be necessary to compute a few d's before t is exceeded. In the next section we shall establish some general results about optimal ordering schemes.

3. OPTIMAL ORDERING IN MATCHING

As already indicated, if we know the expected value of $d(L, L')$ for each L, we should be able to use this information to determine the order in which to compute the d's, and thereby reduce the expected number of d's that need to be computed before the threshold t is exceeded. In particular, we suggested that the d's should be computed in decreasing (more precisely, nonincreasing) order of their expected size. In fact, this is not always the optimal order, i.e., this order does not necessarily minimize the expected number of d's that need to be computed. In this section we present some algorithms for determining the optimal order under various assumptions.

We first give a simple example to show that the decreasing order is not always optimal. Let the labels be the integers 0, 1, and 2, with probabilities $\frac{1}{3}$, ϵ, and $\frac{2}{3} - \epsilon$, respectively, where ϵ is a small positive number. Then the possible values of $d(L, L')$ are 0, 1, and 2, and their probability densities and expected values are as follows:

	Probability of $d(L, L') =$			Expected value
L	0	1	2	of $d(L, L')$
0	$\frac{1}{3}$	ϵ	$\frac{2}{3} - \epsilon$	$\frac{4}{3} - \epsilon$
1	ϵ	$1 - \epsilon$	0	$1 - \epsilon$
2	$\frac{2}{3} - \epsilon$	ϵ	$\frac{1}{3}$	$\frac{2}{3} + \epsilon$

Thus the expected d value is greatest for $L = 0$, next greatest for $L = 1$, and least for $L = 2$. On the other hand, suppose we have already computed a set of d's and have accumulated a value of Σd_i that is between $t - 1$ and t. If we next choose a node with label $L = 0$, to maximize the expected value of d, then with probability $\frac{1}{3}$ we will get $d = 0$, so that the value of Σd_i will still be less than t. On the other hand, if we choose a node with label $L = 1$, then with probability arbitrarily close to 1 we will get $d = 1$, so that Σd_i will be at least t and we can stop.

3.1. The optimal order

The problem of determining the optimal order can be formulated in general terms as follows: Suppose we have a set I_n of n possible operations, denoted by $1, \ldots, n$, each of which produces a nonnegative real value, where the i^{th} operation has cost c_i and its value is the random variable e_i. (In our case, the operations are the computation of the d's, i.e., $e_i = d(\,,\,)$, and they all have the same cost.) Our task is to order the operations so as to achieve $\Sigma e_i \geq t$ at minimum cost Σc_i.

A *state* of the process can be fully characterized by the accumulated value and the set of unused operations. Let the set of all possible states be denoted by S. Clearly $S = \bigcup_{k=1}^{n} S_k$, where S_k is the set of possible states at the k^{th} step of the process. It is easy to see that

$$S_k = \{0, \ldots, E_k\} \times \{\text{all possible subsets of } I_n \text{ of size } n - k + 1\}$$

where E_k is the maximum possible accumulated error at the k^{th} step. Let D denote the set of all possible decisions. Thus $D = I_n \bigcup \{h\}$ where h is the decision to halt the process and i, $1 \leq i \leq n$, is the decision to do operation i. Also let $E[\sigma]$ denote the accumulated error and $A[\sigma]$ the set of available operations corresponding to state σ.

Every state $\sigma \in S$ has the following two decision table entries:

$d[\sigma] \in D$, the optimal decision at state σ.

$c[\sigma]$, the expected cost of a process starting from state σ and making optimal decisions henceforth, i.e., the minimum possible expected cost starting from σ.

We now give a backward induction method for building the optimal decision table, i.e., determining the values of $d[\sigma]$. It is easy to fill in the values of $d[\sigma]$ and $c[\sigma]$ for $\sigma \in S_n$. We then work backwards to fill in the values for the other states.

```
Basis:
    For all $\sigma \in S_n$ do
[Note that there is only one operation (say $i$) available.]
    if $E[\sigma] \geq t$ then $d[\sigma] = h$ and $c[\sigma] = 0$
    else $d[\sigma] = i$ and $c[\sigma] = c_i$
Induction:
[Assuming that we are given $d[\sigma]$ and $c[\sigma]$ for all states
in $S_{k+1}$ we compute them for all states in $S_k$.]
For all $\sigma \in S_k$ do
if $E[\sigma] \geq t$ then $d[\sigma] = h$ and $c[\sigma] = 0$
else
$c[\sigma] = \min_{i \in A[\sigma]} \left( \sum_{\sigma' \in S_{i,\sigma}} p_{i,\sigma}(\sigma') c[\sigma'] + c_i \right)$
    $d[\sigma] = i$ which minimizes the above
```

[where $S_{i,\sigma}$ is the set of states reachable form σ after applying operation i and $p_{i,\sigma}(\sigma')$ is the probability of reaching σ' from σ using operation i. Note that $S_{i,\sigma} \subseteq S_{k+1}$ and also that for all $\sigma' \in S_{i,\sigma}$, $p_{i,\sigma}(\sigma') = \text{Prob}(e_i = E[\sigma'] - E[\sigma])$.]

Theorem 3.1. *The decision table is constructed in this way is optimal.*

Proof: We prove optimality inductively. Clearly, for $\sigma \in S_n$,, the decision $d[\sigma]$ is optimal and $c[\sigma]$ corresponds to the minimum expected cost. Now assume we know the optimal decisions and the corresponding minimum expected costs for all states in S_j, $k+1 \leq j \leq n$. For every $\sigma \in S_k$ we pick the decision which gives the least expected cost. Thus the decision for the current state σ is optimal. ∎

3.2. The optimal static order

In general, the optimal decision at each stage will depend on the outcomes of the previous operations. Thus the above decision algorithm is an adaptive decision strategy. We now consider the subclass of "static" decision strategies, in which the operations are done in a fixed order irrespective of the outcomes of the previously done operations. The optimal static decision strategy is the best one can do in applications where the results of individual operations are not observable.

A static decision strategy for a set of operations can be represented by the ordering that it induces on the set. Let π be an ordering of the elements of I_n For simplicity we will assume from now on that all operations have unit cost.

Theorem 3.2. *The expected cost of the static decision strategy defined by π is*

$$C[\pi] = 1 + \sum_{j=2}^{n} \text{Prob}\left[\sum_{k=1}^{j-1} e_{\pi(k)} < t \right]$$

Proof: Let N be the random variable denoting the number of operations to be performed before halting. Then $C[\pi]$ is given by

$$\sum_{j=1}^{n} j\,\text{Prob}(N = j) = \sum_{j=1}^{n} \text{Prob}(N \geq j)$$

$$= 1 + \sum_{j=2}^{n} \text{Prob}\left[\sum_{k=1}^{j-1} e_{\pi(k)} < t \right]$$

by definition of N. ∎

Corollary 3.3. *Let π^* be the optimal ordering of I_n. Then the ordering $\pi^{*\prime}$ such that $\pi^{*\prime}(j) = \pi^*(j)$ for all $1 \leq j \leq n-1$ is the optimal ordering of $I_n - \{\pi^*(n)\}$.*

Proof: This follows from the additive form of the expected cost derived in Theorem 3.2. ∎

The general idea of our method of determining the best static decision strategy is to compute the optimal ordering for small subsets of I_n and work up inductively to larger subsets. For any subset J of I_n, let $\pi[J]$ be the optimal ordering of J and $c[J]$ its cost. Let the probability density of e_j be f_j; that of $\sum_{j \in J} e_j$ be g_J; and let the probability distribution of $\sum_{j \in J} e_j$ be G_J, so that $G_J(x) = \sum_{j=0}^{x-1} g_J(j)$.

```
Basis:
    For all sets $J \subseteq I$ having just one operation (say $k$)
        do
    [Compute the density function.]
    $g_J = f_k$;
    [Compute the orderings and the costs.]
        $\pi[J](1) = k$ and $c[J] = 1$
Induction:
    [We assume that we have the optimal orderings and
    costs for all subsets of size $j$ and we compute them
    for those of size $j+1$.]
    For all sets $J \subseteq I$ of cardinality $j+1$ do
    [Compute the distribution function.]
        Let $k'$ be any element of $J$.
        $g_J = g_{J-\{k'\}} \otimes f_{k'}$
        where $\otimes$ is the convolution operator.
    [Compute the orderings and the costs.]
        $c[J] = \min_{k \in J} c[J - \{k\}] + G_{J-\{k\}}(t)$
        $\pi[J](j') = \pi[J - \{k''\}](j')$ for $1 \leq j' \leq j$
        and $\pi[j+1] = k''$
        [where $k''$ is the element which minimizes the
        previous expression.]
```

Theorem 3.4. *The above algorithm computes the optimal static decision strategy in $o(n 2^n)$ steps.*

Proof: When the set J has just one element the ordering $\pi[J]$ is trivially optimal and the optimal cost is 1. Now assume that we have the optimal orderings and the corresponding costs for sets of size j. Consider a set J of cardinality $j+1$. The last element of the optimal ordering has to be one of the $j+1$ elements in J. By Corollary 3.3 we know that the remaining elements must also be optimal. Thus we have only $j+1$ possible orderings as candidates for the optimal ordering for J. Of these we choose the one with the least cost as $\pi[J]$. Then evidently $\pi[J]$ is optimal. Note that evaluating the costs involves determining the new probability density function which can be gotten by convolving the probability density of one of the elements of J with the density function of the sum of the remaining elements, which is obtained from the previous inductive step.

If J has j elements, the number steps for computing the minimum and determining the error (including the convolution) is proportional to j. Also the number of sets of size j is $\binom{n}{j}$. Thus the total number of steps is of the form

$$K\sum_{j=1}^{n} j\binom{n}{j} = K'n\sum_{j=1}^{n}\binom{n}{j}\left[\text{since}\binom{n}{j} = \binom{n}{n-j}\right] = o(n2^{n})$$

where K and K' are constants. ■

Note that $o(n2^n)$ is a considerable saving over the brute force search algorithm which involves more than $n!$ steps. However, the algorithm still takes an exponential number of steps. We may not be able to do better as this problem is likely to be NP-hard.

3.3. The decreasing order

The set of operations I_n will be called *orderable* if there is an ordering π such that for all $1 \leq j < k \leq n$ and for all t', $\text{Prob}(e_{\pi(j)} \geq t') \geq \text{Prob}(e_{\pi(k)} \geq t')$. In this section we prove that if this condition holds, then irrespective of the current accumulated error the choice of operation $\pi(j)$ is always better than the choice of $\pi(k)$, i.e., the ordering π is optimal.

Note that there are natural applications where the orderability criterion is satisfied. In many application domains either there is an error or no error, i.e., it may not be possible to grade the errors quantitatively. In these case the random variables e_i take binary values and it can be seen that any such set of operations is orderable. In fact in this case the order is simply the order of decreasing expected error.

Theorem 3.5. *Let the set of operations I_n be orderable and π be the corresponding order. Then the static ordering π is optimal even among adaptive strategies.*

Proof: Our aim will be to show by induction that the optimum decision table of Section 3.1 follows this static ordering. Without loss of generality we can assume that the ordering π is the ascending numerical order, i.e., $\pi(j) = j$ for all $1 \leq j \leq n$. The theorem can then be restated as follows:

T: For all states $\sigma \in S$, if $E[\sigma] < t$ then $d[\sigma] =$ the smallest element in $A[\sigma]$, else $d[\sigma] = h$.

Let us denote the optimal cost at σ by $c[E[\sigma], A[\sigma]]$. It is clear that for all $\sigma \in S$ such that $E[\sigma] \geq t$, $d[\sigma]$ is assigned h by the algorithm of Section 3.1. Therefore we need only consider σ's such that $E[\sigma] < t$. As before, let S_i be the set of states having $n \ \ i+1$ available operations. We have the following cases:

a) If $\sigma \in S_n$, $A[\sigma]$ has only one element. It is easy to see that the basis part of the algorithm assigns this element to $d[\sigma]$. Thus **T** is trivially true.

b) If $\sigma \in S_{n \ 1}$, let $A[\sigma] = \{i,j\}$ and $i < j$. Let $E[\sigma] = t'$ where $t' < t$. Let c_i represent the minimum expected cost given that choice i is made at σ and similarly for c_j. These are the costs computed in the minimization step of the algorithm. Then we have

$$c_i = 1 + \text{Prob}(e_i < t - t')$$

This is clear because the process stops only if the threshold t is exceeded. Similarly

$$c_j = 1 + \text{Prob}(e_j < t - t')$$

From the orderability condition and $i < j$ we have $c_i \leq c_j$. Thus i is an optimal choice at σ, and **T** is true.

c) If $\sigma \in S_j$, $j < n - 1$, we proceed as follows: We assume that **T** is true for all states in $S_{j'}$, $j+1 \leq j' \leq n$, and show that **T** holds for $\sigma \in S_j$. Let $E[\sigma] = t'$, $t' < t$. Let i be the smallest element in $A[\sigma]$ and let j be any other element. We will show that $c_i \leq c_j$ where c_i and c_j are as defined earlier. For convenience we write $A[\sigma]$ as $\{i,j,o\}$ where o represents the remaining elements of the set, and we denote $t - t' - 1$ by τ. Then

$$\begin{aligned} c_i &= 1 + \sum_{x=0}^{\tau} c[t'+x, \{j,o\}]f_i(x) \\ &\leq 1 + \sum_{x=0}^{\tau}\sum_{x'=0}^{\tau-x} c[t'+x+x', \{o\}]f_j(x')f_i(x) \\ &\qquad + \text{Prob}(e_i \leq \tau) \end{aligned}$$

$$\textit{since } c[t'+x, \{j,o\}] \leq 1 + \sum_{x'=0}^{\tau-x} c[t'+x+x', \{o\}]f_j(x')$$

This inequality follows from the fact that $c[t'+x, \{j,o\}]$ is the optimal cost. Similarly

$$\begin{aligned} c_j &= 1 + \sum_{x=0}^{\tau} c[t'+x, \{i,o\}]f_j(x) \\ &= 1 + \sum_{x=0}^{\tau}\sum_{x'=0}^{\tau-x} c[t'+x+x', \{o\}]f_i(x')f_j(x) \\ &\qquad\qquad + \text{Prob}(e_j \leq \tau) \end{aligned}$$

Note that the equality in the last equation is due to our induction assumption that **T** holds for the states in S_{j+1}. We know that i which is the smallest element in $\{i,o\}$ is the optimal choice and hence the equality. Observe that the summations in the last two equations are equal. Thus from these equations and the orderability condition we have $c_i \leq c_j$ for any j in $A[\sigma]$. Thus i is an optimal choice at state σ, so that **T** holds for σ. ■

Corollary 3.6. *If e_i, $1 \leq i \leq n$ are binary-valued then the order of decreasing expected error is optimal.*

3.4. Application to quantitative matching

To apply the results of this section to the quantitative matching of labeled graphs, we need to treat the distance values $d(.)$ [between the labels of corresponding nodes] as random variables. We can do this by considering the set of all possible correspondences between G and H, so that a given node of G, say with label L, can correspond to any (randomly chosen) node of H, say with label L', and $d(L, L')$ becomes a random variable. If we now use an optimal matching order to minimize the expected number of steps needed to exceed the threshold t, then when we try all the correspondences of G with H (in order to find all the t-matches), we will eliminate the mismatches quickly, and thus minimize the expected computational cost of the entire process.

As an important example, suppose that the node labels are real numbers, and that the labels on the nodes of H are randomly assigned with a given probability density. Then the distance between a given label L (on a node of G) and an arbitrary label L' (on a node of H) becomes a random variable, and its probability density can be immediately computed, since for any d we have $\text{Prob}(d) = \text{Prob}\{L' = (L + d)\} + \text{Prob}\{L' = (L - d)\}$. Thus the results of this section allow us to define optimal orderings for the qualitative matching of graphs having random (numerical) node labels.

If G and H are digital images, the node labels are just the pixel gray levels. Thus when an ensemble of images is modeled as a random field, i.e., when we specify the probability density of the gray levels, an image from that ensemble is a special case of a graph having random node labels. In the next section we show how ordered matching reduces the expected computational cost of matching images when we know their gray level probabilities.

4. ORDERED IMAGE MATCHING

In this section we assume that G and H are digital images, where G (the "template") is $m \times m$ and H (the "image") is $n \times n$. Let the gray levels of the pixels in G be denoted by z_i $(1 \leq i \leq m^2)$, and for a given correspondence of G with H, let the gray levels of the corresponding pixels of H be denoted by w_i $(1 \leq i \leq m^2)$. Thus the match error for the given correspondence is $\sum_{i=1}^{m^2} |z_i - w_i|$. We want to order the pixels of G so that, when we compute the terms of the sum in that order, we minimize the expected number of terms that need to be computed before the threshold t is exceeded.

In this section we will primarily be interested in computing the terms in decreasing order of expected difference. Let the image H come from an ensemble in which the probability of gray level w is $p(w)$. [We can estimate the p's by computing the histogram of H; if H is $n \times n$, and there are $h(w)$ pixels with gray level w, then $h(w)/n^2$ is an estimate of $p(w)$.] Then the probability of obtaining difference d for a template pixel with gray level z is $q(d|z) = p(z \;\; d) + p(z + d)$. The expected difference is thus $e(z) = \Sigma dq(d|z)$. Our strategy is to first compute $|z_i - w_i|$ for those z's for which $e(z)$ is greatest, then for those z's for which it is next greatest, and so on, until t is exceeded or until all m^2 differences have been computed.

[As we saw in Section 3, this strategy does not always minimize the expected computational cost (= the number of differences that need to be computed before t is exceeded), though it evidently does maximize the expected value of $\Sigma|z_i - w_i|$ for any given number of terms of the sum. We also saw in Section 3 that if the images are binary, the descending order does minimize expected computational cost. Note that in the binary case, it is easy to compute $e(z)$ explicitly. In fact, let the probabilities of 1 and 0 be p and $1 - p$. The only possible values of d are 0 and 1, and evidently we have $q(1|1) = p(0) = 1 - p$, $q(0|1) = p(1) = p$, $q(1|0) = p(1) = p$, and $q(0|0) = p(0) = 1 - p$. Thus $e(1) = 1 - p$ and $e(0) = p$, so that the descending order simply means that if $p > \frac{1}{2}$, we first compute $|z_i - w_i|$ for all the 0's in the template, while if $p < \frac{1}{2}$ we first compute it for all the 1's.]

The decreasing order may or may not be optimal, but even if it is, the theory of Section 3 does not tell us how much better it is than (say) random order. In the remainder of this section we present experimental results that show the improvement obtained when the decreasing order is used.

We should first point out that there are cases in which using the decreasing order yields no advantage; in fact, there are cases where $e(z)$ is constant for all z, so that no order yields a greater expected rate of increase than any other. Specifically, let the grav level range be $[0,M]$, and let $p(w) = \frac{1}{2}$ for 0 or M, and $p(w) = 0$ otherwise. Then evidently for any z we have $|z - w| = z$ with probability $\frac{1}{2}$ and $M - z$ with probability $\frac{1}{2}$, so the expected value of $|z - w|$ is $e(z) = \frac{z}{2} + \frac{M-z}{2} = \frac{M}{2}$ for all z. At the other extreme, if $p(w) = 1$ for $w = 0$ and $p(w) = 0$ otherwise, then for any z we have $|z - w| = z$ with probability 1, so the expected value of $|z - w|$ is z, which can range anywhere between 0 and M (and similarly if $p(w) = 1$ for $w = M$). [On the other hand, if, e.g., $p(w) = 1$ for $w = \frac{M}{2}$, then $e(z)$ can range between 0 and $\frac{M}{2}$; and similarly for other cases in which $p(w) = 1$ for a specific value of w.] As a final example, let $p(w)$ be constant for all w; then evidently the expected value of $|z \;\; w|$ is $\frac{M}{2}$ when $z = 0$ or M, and $\frac{M}{4}$ when $z = \frac{M}{2}$, while it takes on intermediate values for other z's, so that it ranges between $\frac{M}{4}$ and $\frac{M}{2}$. These examples suggest that using decreasing order is most advantageous when $p(w)$ is sharply peaked at a single value; also somewhat advantageous when $p(w)$ is uniform; and not advantageous when $p(w)$ has two equal, widelv separated peaks.

These observations are borne out by the examples shown in Figures 1 5. In all of these figures both the image and the template have 16 gray levels, and the size of the template is 16×16. The image and template are both randomly generated to have (approximately) given histograms. Each figure shows the advantage of decreasing order over random order in terms of the fraction of total mismatch.

In Figures 1 and 2 $p(w)$ is uniform; in Figure 1 the template also has a uniform gray level probability density while in Figure 2 its probability density is peaked at a single value near the center of the gray level range. In both cases, ordered matching is advantageous, but the advantage is not great even in Figure 1, and it is negligible in Figure 2. This is because in the Figure 1 case the template has many pixels at the ends of the gray level range, where the expected difference value is $M/2$, and when we do ordered matching we can compute these differences first to get a relatively fast increase in the mismatch; but in the Figure 2 case the template pixels are nearly all in the middle of the gray level range, so that in both ordered and random matching we have no choice but to use them. To estimate the advantage of ordered over random matching in the Figure 1 case, suppose t corresponds to 40% of the total mismatch. Then if we use random order, we cannot expect to exceed t until 40% of the pixels have been compared; but if we use decreasing order, we can expect to exceed it when about 30% of the pixels have been compared. Thus even in this case, ordered matching reduces the expected computational cost by 25%.

As expected, the advantages of ordered matching are greater in the case shown in Figure 3, where the template has a uniform gray level probability density, but the image's density is peaked at the middle of the range. Here, if we use decreasing order, we exceed $t = 40\%$ when less than 25% of the pixels have been matched; but if we use random order, we do not exceed it until 40% of the pixels have been matched. Thus the cost saving in this case is nearly 50%.

The saving is ever more impressive in the case shown in Figure 4. Here the image and template both have densities that are peaked in the middle of the range. Thus when we use random order, the differences are low; but when we use decreasing order, we first compute the differences for those template pixels that are near the ends of the range, so that we rapidly accumulate high differences. In this example, when we use decreasing order we exceed $t = 40\%$ when only about 17% of the pixels have been matched, but when we use random order we need to match over 40% of the pixels, so that the cost saving is about 60%. Note that, as we see in Figure 5, when the image and template both have peaked densities but the peaks are far apart, the advantage of using decreasing order is not great; here the random order also leads to a rapid accumulation of mismatch, since the image and template gray levels tend to be far apart.

These results illustrate the computational savings that can be obtained by taking advantage of knowledge about the gray level probabilities in the image (and template). In the next section we will show how knowledge about the spatial arrangement of the gray levels (in particular, about probability densities of local property values) can be used to yield further improvements.

5. LOCAL PROPERTY MATCHING

As we saw in Section 4, for some types of gray level probability densities, there is little or no advantage to using ordered matching. In particular, if the image contains primarily low and high gray levels, $e(z)$ is approximately constant for all z. We can sometimes improve this situation by computing a suitably chosen local property at each pixel of the template and of the image. This converts the image into a local property array which may have a more nearly unimodal probability density of values. We can then replace the original matching problem by the problem of matching the local property arrays derived from the template and the image, and ordered matching can be advantageously used for this new problem.

We can obtain a local property array that is more nearly unimodal than the original image by taking advantage of local gray level dependencies in the image. In other words, if we know the second order gray level probabilities for the image (i.e., the joint probabilities of given pairs of gray levels occurring in given relative positions), we can compute functions of the gray levels—in particular, local properties—that can be expected to have, say, high values. In this section we give several illustrations of how this approach can be used to generate derived arrays in which ordered matching is more advantageous than it was for the original images.

It should be realized, of course, that finding good matches to a local property array is not equivalent to finding good matches to the original template. If a given correspondence yields a good match between the template and the image, it also yields a (relatively) good match between the local property arrays derived from them. [For example, suppose the local property is of the form $f'(x,y) = \sum_{i=1}^{k} c_i f(u_i,v_i)$, where the f's are gray levels and the u_i,v_i are neighbors of (x,y). If $|g(u,v) - h(u,v)| < \alpha$ for all (u,v), where g and h denote the pixel gray levels in G and H, respectively, then $|g'(x,y) - h'(x,y)| = \left| \sum_{i=1}^{k} c_i \left(g(u_i,v_i) - h(u_i,v_i) \right) \right| \leq \sum_{i=1}^{k} c_i |g(u_i,v_i) - h(u_i,v_i)| < \alpha \sum_{i=1}^{k} c_i$.] On the other hand, a good match between the local property arrays does not imply a good match between the template and the image. [As a very simple example, let the local property be defined by $f'(x,y) = f(x+1,y) - f(x,y)$, and suppose that $g(u,v) = h(u,v) + c$ for all u,v; then $g'(x,y) = h'(x,y)$ for all x,y (since the c cancels), so the local property arrays match perfectly, whereas the original image and template have a mismatch of $\sum_{u,v} \left(g(u,v) - h(u,v) \right) = m^2 c$.]

It may nevertheless be advantageous to find good matches between the derived property arrays of the image and template. These matches are guaranteed to include the good matches between the image and template themselves. If good matches are rare (as is usually the case), it is inexpensive to test the correspondences that give good matches between the property arrays to verify which of these, if any, give good matches between the image and template. At the same time, by using ordered matching on the derived arrays to reduce the expected computational cost, we may have greatly decreased the average amount of computation that needs to be done at every pixel.

The advantage of matching local property arrays is illustrated in Figures 6–8. In all of these figures, the template and image both have four gray levels with (approximately) equal probabilities (i.e., their gray level probability densities are uniform). [We recall that in this case the advantage of using ordered matching was not great.] Figure 6 shows two such images, in one of which the joint probabilities of similar gray levels are high for pairs of pixels that are horizontal neighbors, while in the other they are high for vertically neighboring pairs. The local property used is $f'(x,y) = f(x+1,y) - f(x,y)$. (Note that the local property arrays have both positive and negative values.) The template (of size 16 × 16) is a portion of the image in Figure 6a; in Figure 7 we show results when it is matched to another portion of the same image. Here the histograms of the property value arrays both have sharp peaks in the middle of the value range (i.e., at 0), and as expected from the case of Figure 4, the saving resulting from using the decreasing order is substantial. On the other hand, Figure 8 shows results when the same template is matched to a portion of the image in Figure 6b. Here the histogram of the property value array obtained from the image is much flatter, and as a result, the saving is not as great (compare Figure 3).

A final example is shown in Figures 9–10. Here the image and template both consist of pure "salt-and-pepper noise", i.e., two gray levels ("black" and "white") occurring with equal probabilities. This means that their histograms consist of two spikes of equal height located at the ends of the gray scale, so that ordered matching should yield no advantage at all; this is confirmed by the graphs in Figure 9. In this case, whether we use ordered or random matching, to exceed $t = 40\%$ we must match 40% of the pixels. The improvement is dramatic when we replace the image and template by local property arrays; in this case we have used the local property

$$f'(x,y) = \max[|f(x+1,y) - f(x,y)|, |f(x-1,y) - f(x,y)|,$$
$$|f(x,y+1) - f(x,y)|, |f(x,y-1) - f(x,y)|]$$

Evidently, $f'(x,y)$ will have value 0 with probability about $\frac{1}{16}$, and value 1 with probability about $\frac{15}{16}$. Thus its histogram has a major peak at the high end of the gray scale (and a much smaller one at the low end), so that ordered matching should be very advantageous. This is borne out by the graphs in Figure 10, where we see that using ordered matching allows us to exceed $t = 40\%$ when only a little over 5% of the pixels have been matched—an improvement by a factor of nearly 8.

The results of this section show that knowledge about second order gray level probability densities, or about the (first-order) probability densities of local property values, can yield further reductions in the expected computational cost of matching. It will be recalled that these types of probabilistic information are commonly used to characterize image textures [2]. Thus this section illustrates how we can reduce the expected cost of image matching if we know the textures of the images that are being matched.

In the example given in this section, the local property used was simply a directional difference of gray levels. Other local properties could be used, as appropriate, in other cases. It is of interest to recall [3] that matching of image differences (or derivatives), rather than the images themselves, may yield an optimal signal-to-noise ratio (between matches and mismatches) when the images are "busy". Our results show that matching of differences may have additional advantages in reducing the expected computational cost of the matching process.

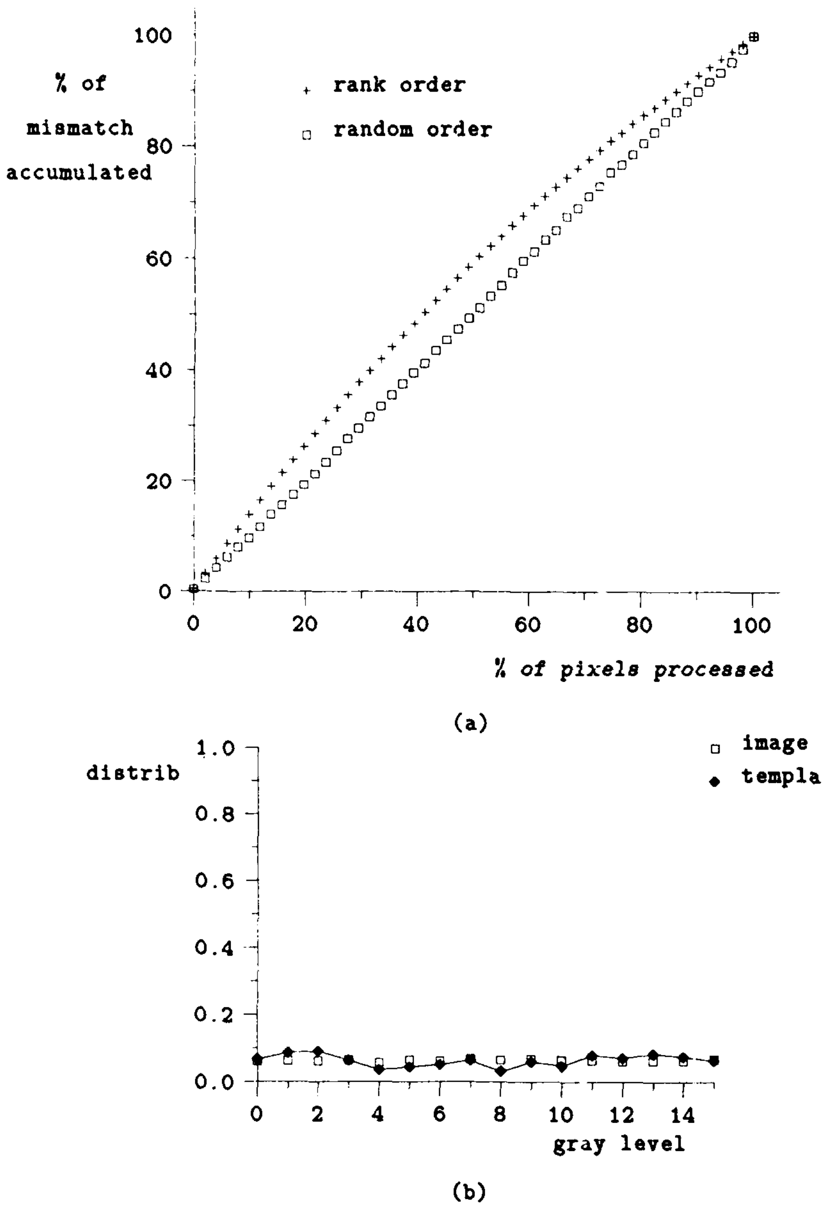
6. CONCLUDING REMARKS

The basic idea of using ordered matching to reduce expected cost was proposed by Nagel and Rosenfeld over 15 years ago [4]. This paper has developed a general theoretical framework for this approach, and has also shown how to obtain further improvements by matching local property arrays. We have seen that substantial cost savings (sometimes of 50% or more) can be obtained in this way. As shown in [4], the savings can more than compensate for the initial cost of ordering the template pixels.

Other methods of reducing the cost of image matching have been proposed—for example, methods that make use of multiple-resolution representations of the images. We plan to investigate the extension of our approach to the multiresolution case in a subsequent paper.

REFERENCES

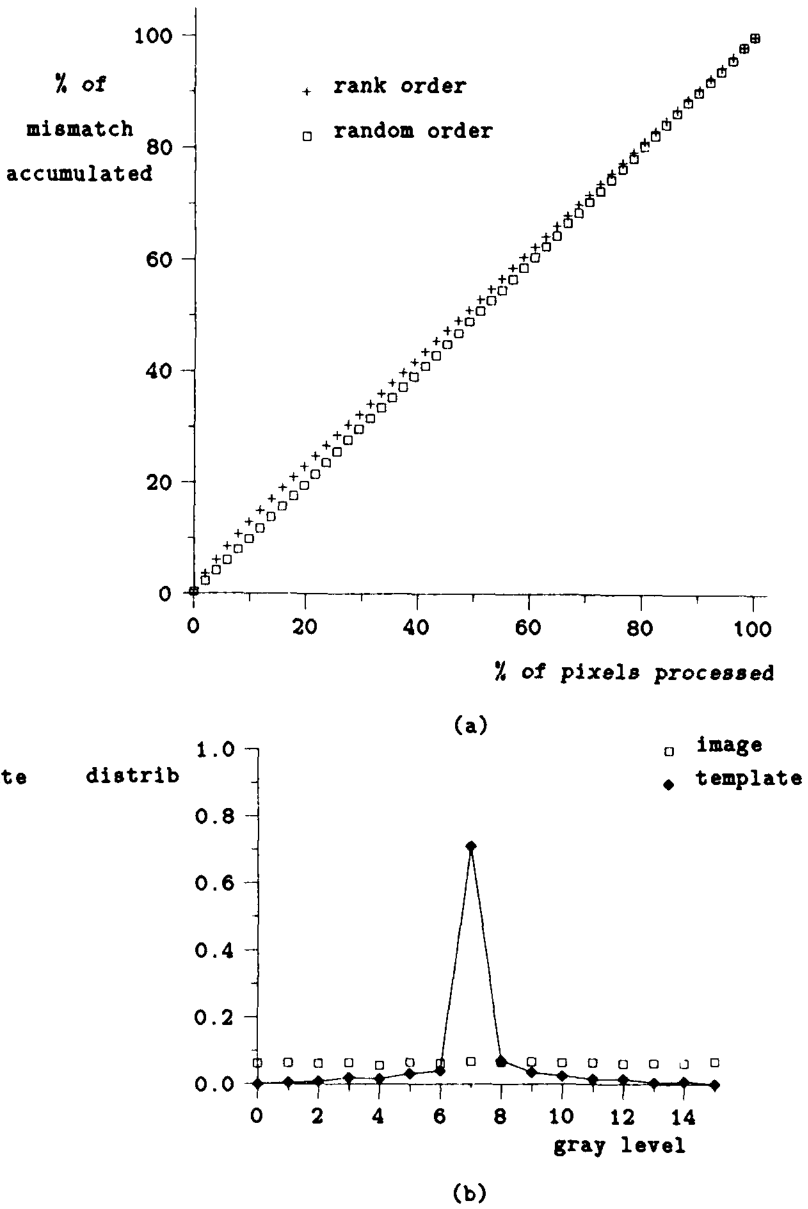
1. A. Amir, Multidimensional and relative approximation pattern matching, UMIACS-TR-87-41, CS-TR-1912, September 1987.
2. A. Rosenfeld and A.C. Kak, *Digital Picture Processing*, second edition, 1982, Section 12.1.5.
3. A. Rosenfeld and A.C. Kak, *Digital Picture Processing*, second edition, 1982, Section 9.4.2.
4. R.N. Nagel and A. Rosenfeld, Ordered search techniques in template matching, *Proc. IEEE* **60**, 1972, 242–244.



(a)

(b)

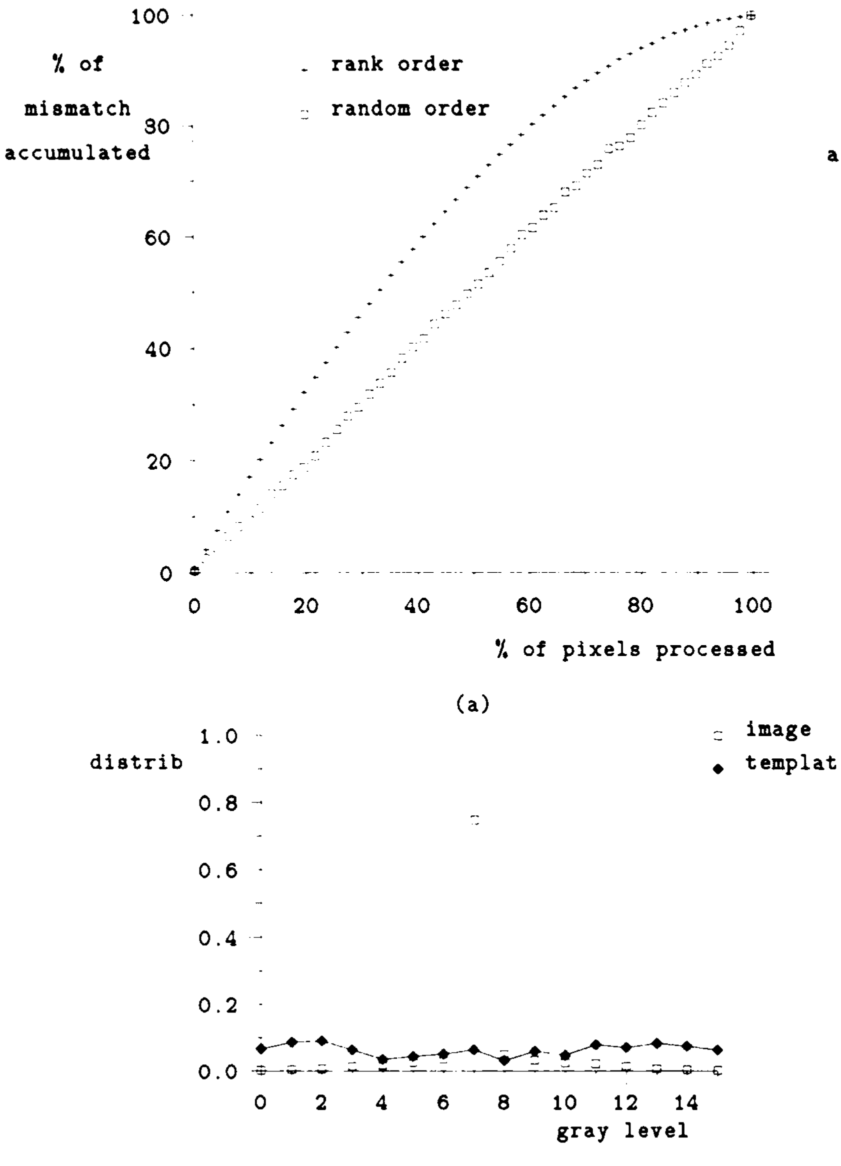
Figure 1



(a)

(b)

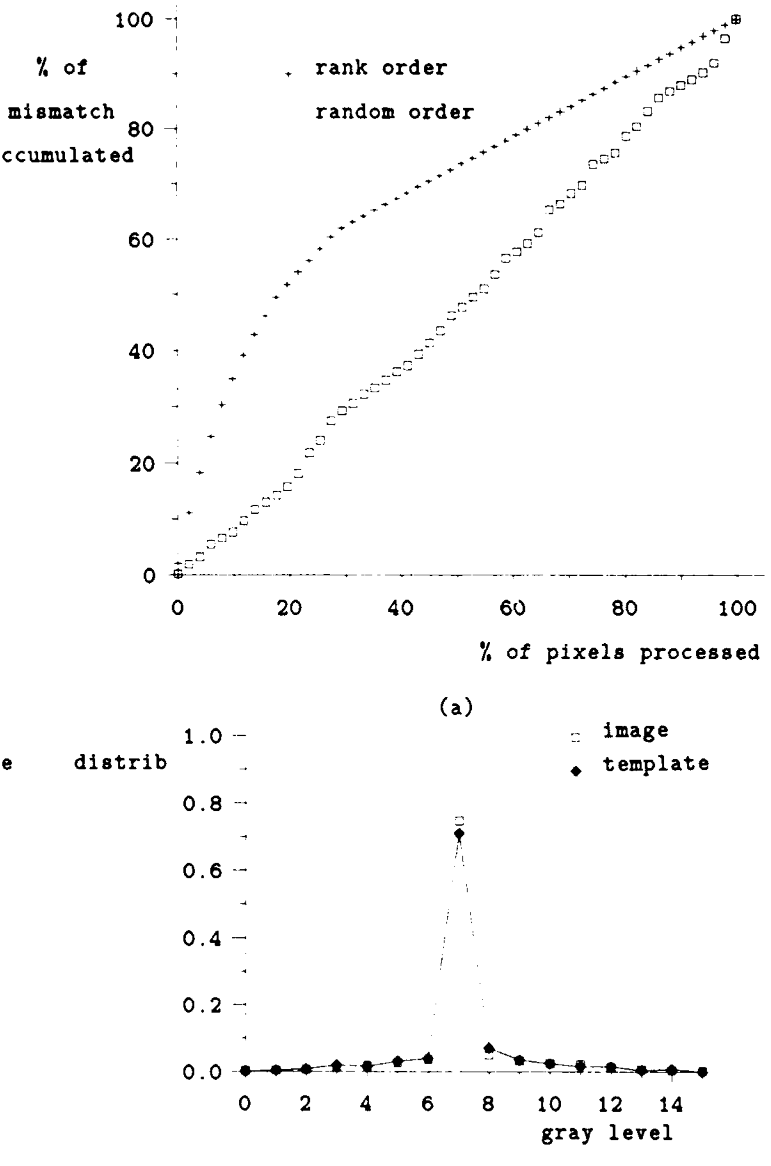
Figure 2



(a)

(b)

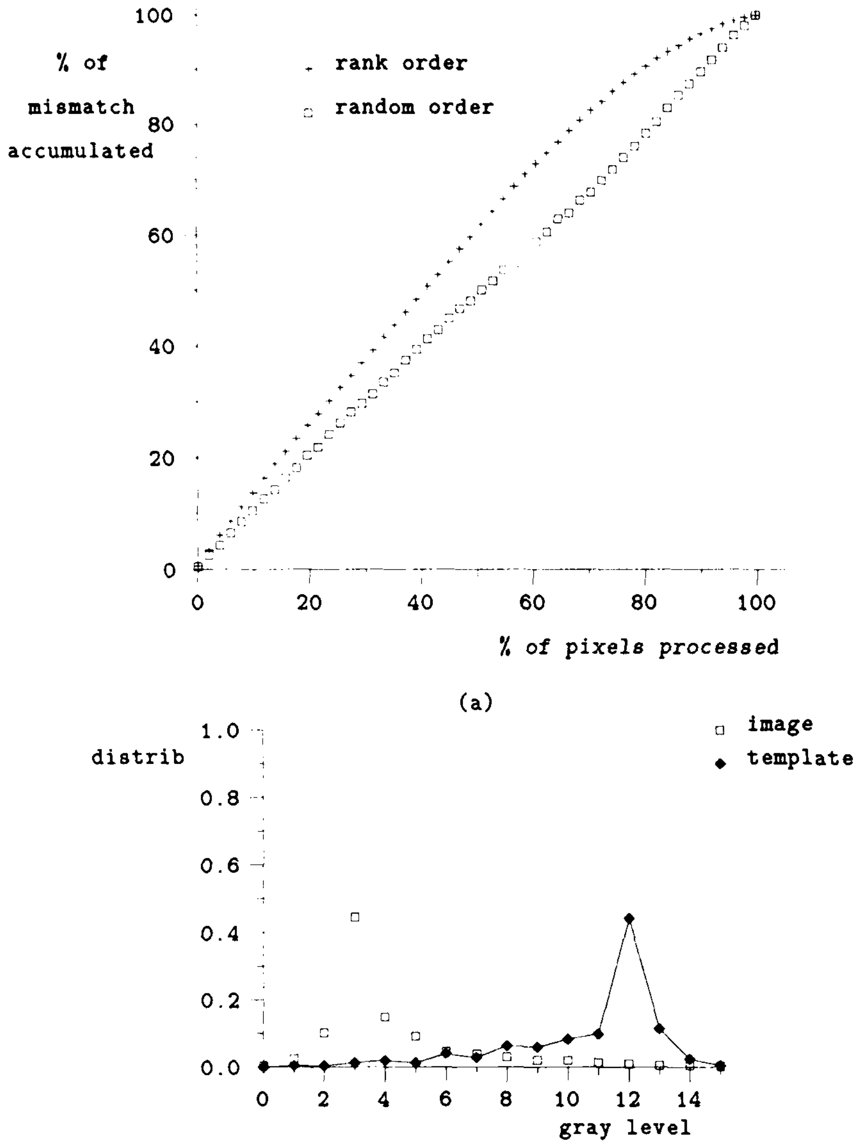
Figure 3



(a)

(b)

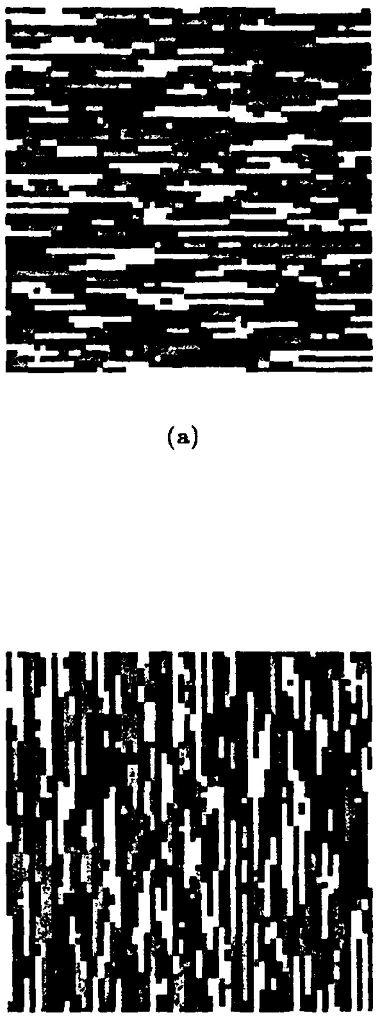
Figure 4



(a)

(b)

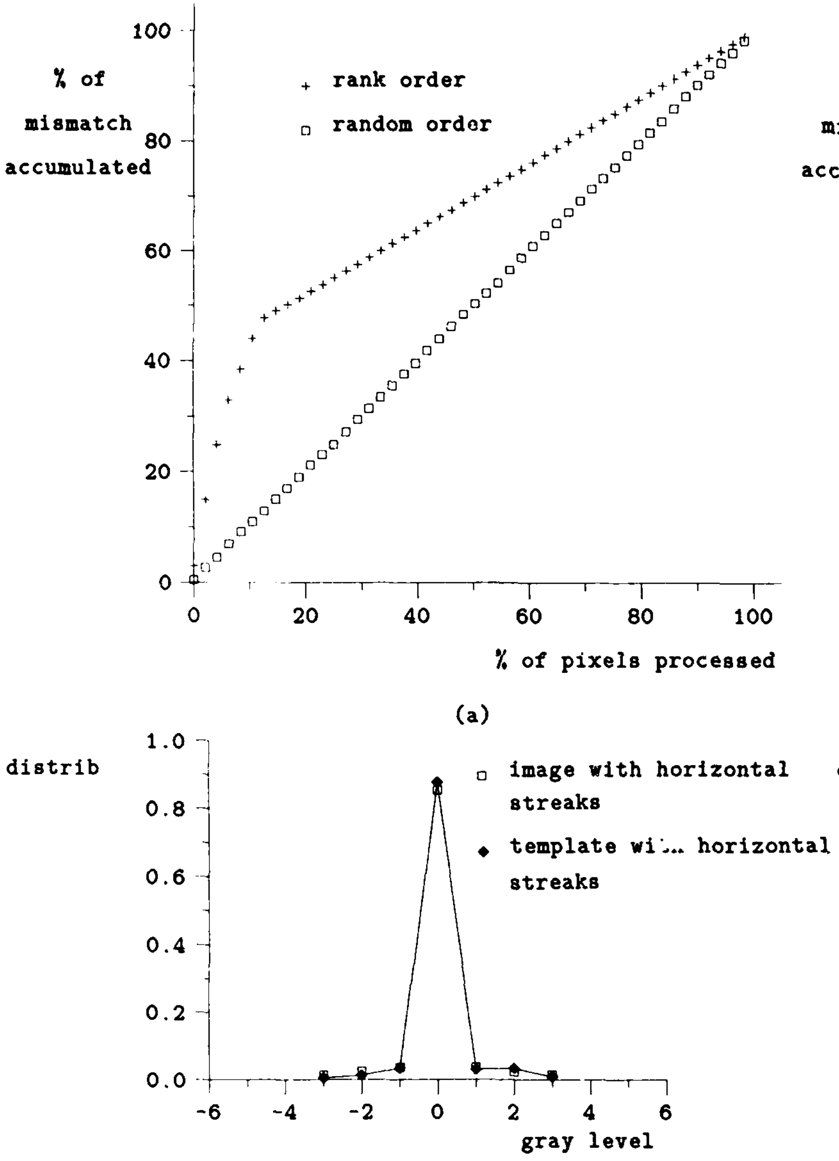
Figure 5



(a)

(b)

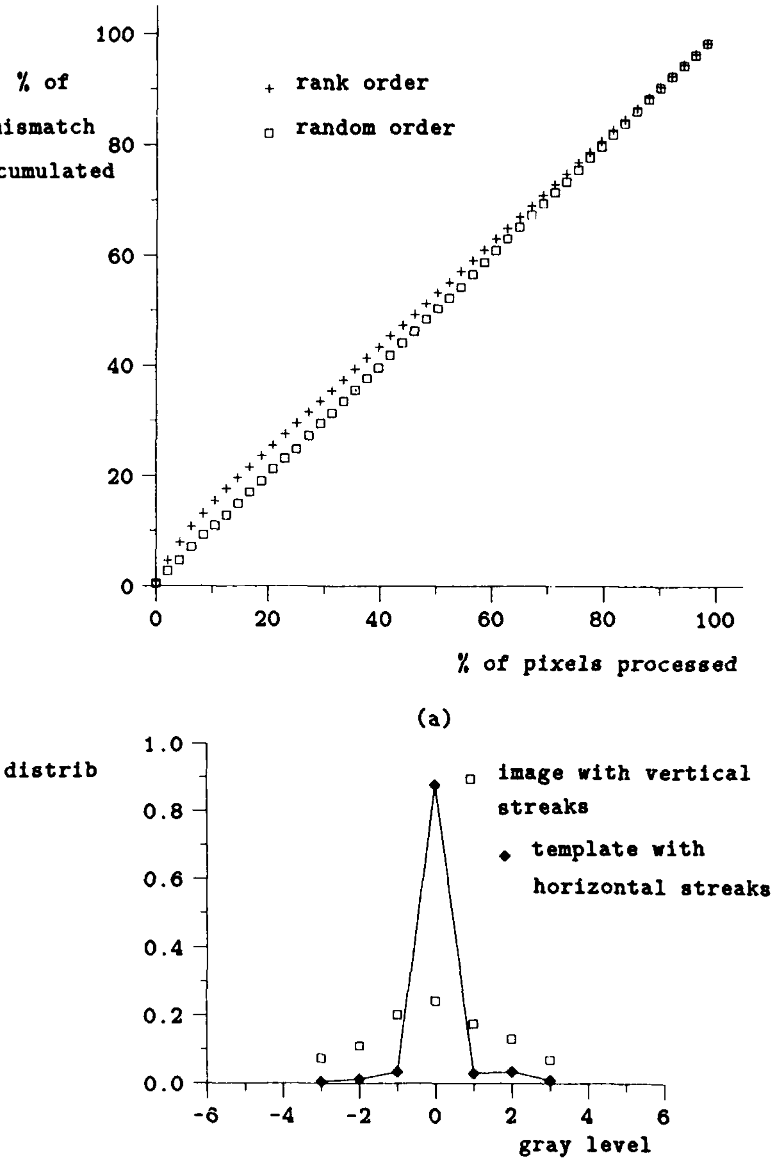
Figure 6



(a)

(b)

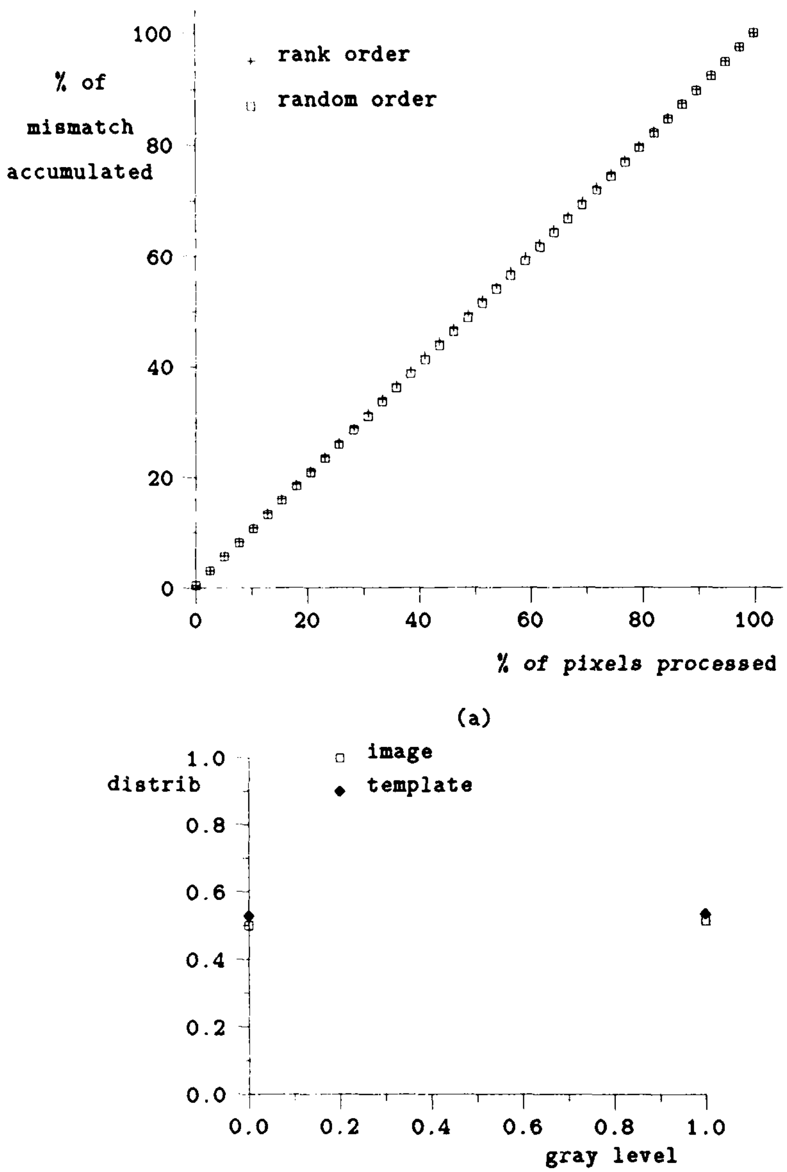
Figure 7



(a)

(b)

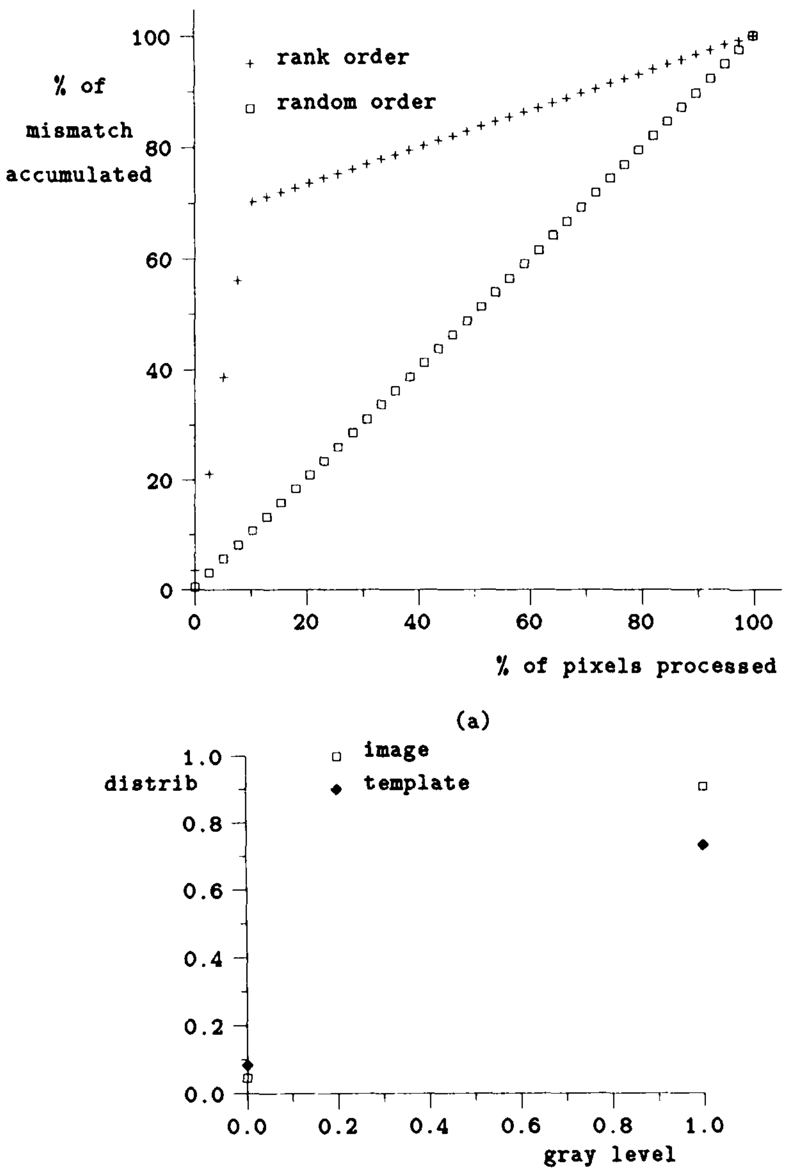
Figure 8



(a)

(b)

Figure 9



(a)

(b)

Figure 10

Object Recognition from a Large Database Using a Decision Tree

Michael Swain

Department of Computer Science
University of Rochester
Rochester, NY 14627
swain@cs.rochester.edu

Abstract

Decision trees have been used for many years in pattern recognition systems. However, their use in visual recognition has been restricted to the pattern recognition domain, where features are required to be measurements which form the axes of a vector space, and which are often assumed independent. Computer vision researchers have studied the use of more powerful recognition techniques that consider object topology and viewpoint consistency, among other things. How to structure a large database for efficient recognition when using these techniques has not been well studied. Here I claim that the decision tree data structure can be employed as successfully outside of the pattern recognition domain as it has been within. To substaniate my claim I describe a system for recognition from a large database that recognizes polyhedra in crowded scenes and uses object topology, comparisons of length and angle, and viewpoint consistency as tests.

Given the a priori probability of polyhedra to appear in the image, viewpoints from which they are seen, and the errors which occur in detecting their edges, the decision tree is automatically constructed from the database using the criterion of minimizing the expected recognition time. Since the optimal solution of this problem is intractible, a greedy method is used based on minimizing the expected entropy of the a posteriori distribution of possible matches. The motivation for the entropy measure is derived.

Since object topology is a major source of information for the recognition system, the space of viewpoints is divided according to principal view regions. The large number of regions that this results in does not reduce the run-time efficiency of the system, since any number of regions can be considered at one time by the recognition system, whose state is determined by a position in the decision tree. Other measurements are also used, for example angle measurements and comparisons of length, where they will provide more information than the topological data. The viewpoint consistency constraint is used to provide an exact viewpoint determination as soon as is economical from information-theoretical point of view. By considering matches directly to the model at run-time instead of principal views the redundancy of the principal view representation is avoided and the viewpoint consistency constraint can be efficiently applied.

1 Introduction

The human visual system can recognize objects from an enormous database in real time. Many vision systems operate under real-time constraints, and a vision system operating in a fairly unconstrained environment may have to recognize objects from a database that is an appreciable fraction of the size of the human database. Large databases have been considered by Pattern Recognition researchers, who have achieved success at indexing into them using decision trees [Wang and Suen 1984]. Under the Pattern Recognition paradigm the features form the axes of a vector space and an object is represented by a point in this feature space [Fu 1968]. The features are often assumed independent.

The recognition techniques considered here are object topology, local quantitative measurements and the viewpoint consistency constraint. They do not fit naturally into the Pattern Recognition paradigm, but this does not prevent a vision system that uses them to structure its database in the form of a decision tree.

Given a probability distribution on the frequency of objects, the viewpoints from which they are seen, and the errors which occur in the low-level system, a decision tree can be built that optimizes the expected cost of recognition. Thus, a decision tree data structure differentiates among more important and less important features for distinguishing among objects. This difference in importance of features cannot be determined a priori but is instead a function of the database. Turney *et al.* [1985] refer to this concept as *saliency*.

The aim of this work is to show that a decision tree should be considered as a data structure to store a database for fast visual recognition outside of the field of pattern recognition. Here a recognition system is described, called the Tree system, that uses a decision tree. For simplicity, the objects are restricted to be polyhedra, but it is expected that the overall design of the system will extend to less restricted worlds. This is because the tests used do not have to be restricted to the ones used in the Tree system, but tests for closure, termination, inside-outside, color and texture could be used as well, for example.

Because the Tree system uses object topology as a major route to recognition, the concept of *principal views* [Koenderink and van Doorn 1976] [Freeman and Chakravarty 1980] is used in the construction of the decision tree. Each principal view defines a family of possible projections, all members of which are topologically identical and differ only by a continuous transformation. The Tree system uses principal views in the preprocessing stage to calculate the possible topological appearance of the objects in the database. The decision tree itself, however, is almost completely independent of them, and instead deals directly in matches to the object model. This was done to avoid the problems associated with the naive application of principal views, which are

1. The correspondence among the same edges in different principal views is not represented.

2. There are a large number of principal views. To solve for the viewpoint it is often not necessary to discover the exact principal view that occurs in the image.

Because the decision tree deals directly in matches to the model, states in the decision tree correspond to matches to object edges, and so problem 1) does not exist. Problem 2) is eliminated by allowing the possibility of solving for the viewpoint before the principal view in the image has been uniquely determined. Instead it is done when its utility rises above the utility of testing for more topological constraints or for other viewpoint invariant features. In the Tree system principal views provide only the lowest grain to which the possible viewpoints are partitioned; at any point in the tree the possible range of viewpoints consists of a collection of principal views of one or more models.

2 Previous Work

The theory of the use of decision trees in sequential pattern recognition is covered in [Fu 1968]. An application to a large database of two-dimensional objects can be seen in [Wang and Suen 1984].

Burns and Kitchen [1987] have designed and implemented a system with a similar approach to the one described here. Rather than constructing a decision tree as the Tree system does, they construct what they call a prediction hierarchy. They have a set of rules for building the tree whose aim is to minimize the size of the hierarchy, rather than to minimize run-time. At run-time the hierarchy is searched from the bottom up by combining features found in the image whenever the hierarchy allows it. They do not consider the run-time complexity, nor does the design explicitly aim to achieve the minimum complexity, but since the search can benefit from parallel processing it could be competitive.

Cooper and Hollbach [1987] have suggested using a series of filters arranged in a hierarchy, using local computationally inexpensive tests in parallel over the image and principal view database at the beginning and more computationally expensive tests at later stages when the number of possible models has been substantially reduced. The optimal decision tree approach, by contrast, can eliminate models without explicitly testing against each of them, and is constructed to use the features that are most salient, as dictated by the database. The decision tree approach cannot benefit from massive parallelism, but does the best possible without it and does not suffer from its costs.

Robert Bolles has designed systems for fast recognition for robotics that use preprocessing to design an optimal strategy at runtime [Bolles and Cain 1982] [Bolles and Horaud 1986]. His approaches only match against one model at a time, and therefore rely on the uniqueness of the features being detected to achieve fast runtime. Chris Goad [1987] has also investigated using preprocessing to minimize recognition time. His approach can only consider one possible assignment of image features to model features at a time, and so will have complexity linear in the size of the database. The decision tree has the potential of logarithmic time complexity.

3 Decision Tree Construction

The standard way of building an optimal decision tree is to build it back-to-front using a dynamic programming approach [Fu 1968]. The tree is built back-to-front because the optimal decision at a particular node depnds on knowing all future optimal decisions. Building an optimal decision tree is intractible for the domain the Tree system is designed for because the space of possible states is large.

Instead of constructing an optimal decision tree, one can use a greedy method to construct the tree in a forward fashion. An estimate is made of the expected cost of recognition in the subtrees that would result from all the possible tests at a given leaf of the partial decision tree. Based on this estimate, the test is chosen. In the appendix it is shown that in many cases the estimate takes the form of an information entropy function (see [Shannon 1948] or a text in information theory [Jones 1979]). The examples in Section s5 assume an entropic cost function. To improve the estimate lookahead can be used, in a similar fashion to game tree search [Slagle and Lee 1971]. The 'opponent' in this case is nature, or chance.

4 Dealing With Errors

It is possible to account for errors in the input to the recognition system either in the construction of the decision tree or at runtime. If errors are considered in the construction of the decision tree, the nodes in the decision tree represent states which include the possibility that an error has been made. The same object becomes a possible match on more than one branch of the tree, a condition termed overlap in the Pattern Recognition literature [Wang and Suen 1984].

If the runtime approach is used, as is used in the Tree system, the states do not include the possibility of error. Instead multiple processes explore the decision tree concurrently. If doubt in a decision rises above a threshold, processes simultaneously investigate the branches of the decision tree that descend from each plausible outcome. The number of processes is kept to within the number of processors available by ranking them according to confidence. The confidence associated with initial thread of control, C_1, is initially set to 1. Whenever decision i is made, the confidence of the thread of control is multiplied by the probability, P_i, that the decision made was the correct one. So the confidence of the i'th thread of control after its k'th decision is

$$C_i(k) = \prod_{j=1}^{k} P_j$$

With C defined as described, it corresponds to the probability that the thread of control is the correct one, under the assumption that the probabilities of each decision being correct are independent.

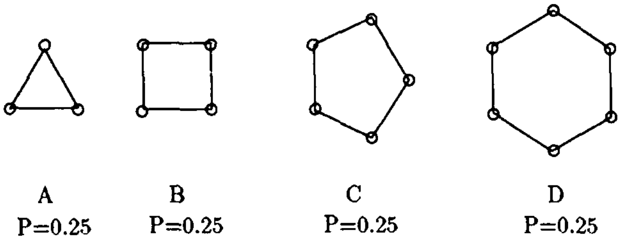


Figure 1: Model Base (Example 1)

Processes are terminated either when they fall below a confidence threshold or when they arrive at a leaf node in the decision tree. The confidence threshold can be made dynamic at the cost of some communication among the processors. A dynamic threshold is desired because not all objects are necessarily of the same probability, and low-probability errors may occur, pushing the confidence of all the processes downward. A list of the most promising terminated processes can be maintained to be reactivated if the confidence of running processes drop.

5 Examples

This section describes two examples that demonstrate various aspects of the recognition system.

The first example builds a complete decision tree for some very simple objects. An interesting feature of the this example is that a greedy technique would not guarantee an optimal tree, despite its simplicity.

The second example builds part of a decision tree for one object that is significantly more complex than those in the first example.

5.1 Example 1

The objects are a triangle, a square, a pentagon and a heptagon. They are shown in Figure 1. All edges are the same length in each model. The models are all assumed equiprobable, so each may appear with probability 0.25. The range of possible viewpoints is such that (scaled) orthographic projection is a good approximation, and so parallel lines appear parallel and lines of equal length in the models appear equal length in the image.

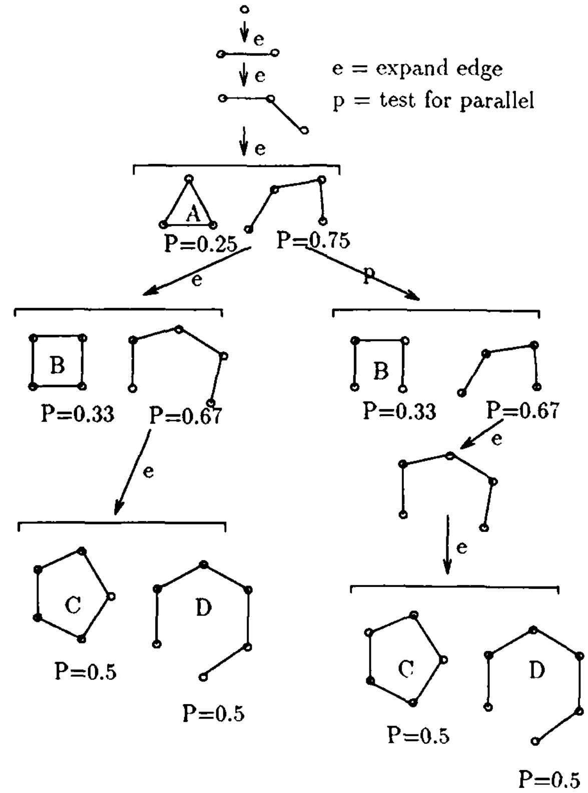


Figure 2: Two Possible Search Trees (Example 1)

Figure 2 shows all possible search trees. Since the objects are so simple there is only one point at which a choice can be made. The choice is whether to (1) expand the fourth edge or (2) test the first and third line segments for parallelism. As is evident, the choice to expand the fourth edge is the better one. Let us calculate the expected entropy of both choices.

We are at state i, which is reached with probability 0.75. For the choice of expanding the fourth edge, we find two possible outcomes. In the case when the object is a square the cycle of edges closes (state ii). Both for the case of the

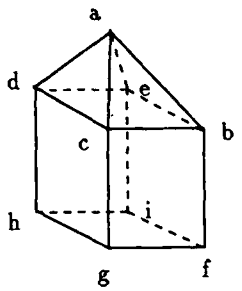


Figure 3: The House Model

pentagon and the heptagon the cycle is not closed (state *iii*). The probabilities of each state, given that we have reached state *i* are

$$\begin{aligned} P(ii|i) &= \frac{.25}{.75} = 0.333 \\ P(iii|i) &= \frac{.5}{.75} = 0.667 \end{aligned}$$

The entropy of state *ii* is zero, since the match is known. The probabilities each match once state *iii* has been reached are $P(\text{pentagon}) = 0.5$ and $P(\text{heptagon}) = 0.5$, and so

$$\begin{aligned} H(iii) &= -\sum_j P_j \log P_j \\ &= -2(0.5 \log 0.5) \\ &= 0.693 \end{aligned}$$

The expected entropy is

$$\begin{aligned} E(H_1) &= (0.333)(0) + (0.667)(0.693) \\ &= 0.462 \end{aligned}$$

A similar calculation shows that the expected entropy of choice 2 is equal to the expected entropy of choice 1 and so the greedy method of minimizing entropy does not guarantee choosing the optimal tree in this case. Searching ahead one ply would enable the entropy method to choose the correct tree tree, however.

5.2 Example 2

Figure 3 shows the object whose pose the decision tree will be designed to recognize (the 'house' figure from [Burns and Kitchen 1987]). Figure 4 shows the principal views chosen a priori to have greater than zero probability of occurence. I will ignore testing for parallelism, length of lines or solving for the viewpoint and only consider the choice of edge to expand. This will be done using the greedy entropy measure. Figure 5 shows possible search trees starting from a vertex of degree 4 and assuming the first edge expanded finds that the vertex of degree 4 is adjacent to another vertex of degree 4.

Because of the symmetries of the 4–4 pair of vertices there are only two different choices of edge to expand which I will call *end* and *side* (left and right branches respectively in Figure 5). Tables 1 and 2 summarize the relevant information for the *end* and *side* choices respectively. In these tables the

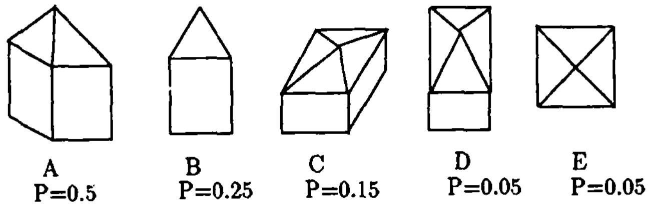


Figure 4: Some Principal Views of the House

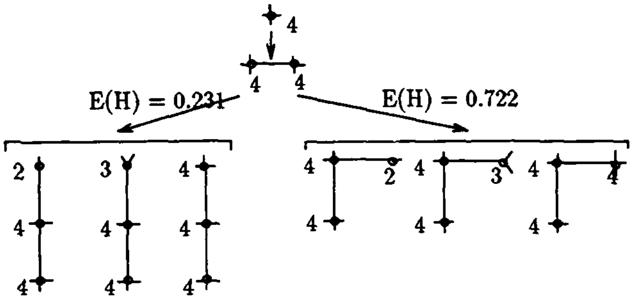


Figure 5: Possible Search Trees (Example 2)

views columns give the number of times the graph or match occur in each principal view. The frequency f can be calculated from the information in the *views* column. For a graph or match G it is just

$$f(G) = \sum_{v \in V} n_v(G) P(v)$$

where V is the set of principal views, $n_v(G)$ is the number of occurences of G in principal view v, and $P(v)$ is the probability of the principal view v. Note that $n_v(G)$ may be greater than one for a match G because of symmetry. The probabilities are obtained from the frequencies are by normalization. The entropy is

$$H(G) = -\sum_{i \in S} p_i \log p_i$$

where S is the sample space under consideration. In our case the sample space is the set of possible matches. The expected entropy is

$$E(H) = \sum_G P(G) H(G)$$

where the sum is over all possible graphs that are obtained by expanding the edge being considered.

The tables show that the expected entropy is lower one chooses to expand the *end* edge ($E(H) = 0.231$) over the *side* edge ($E(H) = 0.722$). Therefore, using the minimum entropy method without lookahead to construct the tree one would choose to expand the end edge.

6 Conclusion

The Tree recognition system described in this paper is being implemented to test the growth in size of the decision tree with the size of the database, its robustness to errors in input and to inaccurate priors, and its runtime complexity. We are

graph	views	prob.	match	views	prob.	entropy
4–4–2	2C 2D	0.444	a–b–f	2C 2D	1.0	0.0
4–4–3	4D	0.222	b–a–d	2D	0.5	1.04
			b–c–d	1D	0.25	
			d–c–b	1D	0.25	
4–4–4	2C	0.333	b–a–d	2C	0.3	0.0
Expected entropy: 0.231						

Table 1: Expand *end* edge

graph	views	prob.	match	views	prob.	entropy
4–2 4	1D	0.056	c–b–f	1D	1.0	0.0
4–3 4	4C 2D	0.778	c–a–b	2C 1D	0.5	0.69
			a–b–c	2C 1D	0.5	
4–4 4	3D	0.167	c–a–b	1D	0.333	1.10
			a–b–c	1D	0.333	
			b–c–a	1D	0.333	
Expected entropy: 0.722						

Table 2: Expand *side* edge

also looking into an real-time incremental decision tree construction algorithm, which would allow the system to collect its priors from experience and adjust its recognition strategy on-line. The off-line procedure described here requires statistics to be collected and the decision tree constructed prior to runtime.

Acknowledgements

I would like to thank my advisor Dana Ballard for his guidance and suggestions, Paul Cooper and Nancy Watts for influential discussions, and Chris Brown for his encouragement and support.

This work was supported in part by U. S. Army Engineering Topographic Laboratories research contract no. DACA76-85-C-0001, in part by NSF Coordinated Experimental Research grant no. DCR-8320136, in part by ONR/DARPA research contract no. N00014-82-K-0193 and in part by an Natural Sciences and Engineering Research Council of Canada Postgraduate Scholarship.

A Entropy as a Measure of Expected Cost

Since the optimality criterion being used for decision tree construction is minimum expected search depth, the best evaluation function to use for constructing the tree is one that estimates the expected search depth from that node. Here we show that under many conditions that function takes the form of an appropriately scaled entropy measure.

Theorem 1 *Suppose every test has m equiprobable outcomes and that each of the n possible events falls into exactly one of the outcomes of each test. If every test has cost c then the expected cost C of the search is*

$$E(C) = -c\sum_{i=1}^{n} p_i \log_m p_i$$

with the understanding that

$$p_i \log_m p_i = 0$$

if $p_i = 0$ or 1.

Proof. The proof is by recursion.

$E(C) = 0$ at a leaf node. Since at a leaf node $p_j = 1$ for some j and $p_i = 0$ for all $i \neq j$ then

$$\sum_{i=1}^{n} p_i \log_m p_i = 0$$

by definition and the theorem holds.

For a node that is not a leaf assume the theorem holds for all its k subtrees. Then the expected cost of the search is

$$E(C) \;=\; c + \frac{1}{m}\sum_{j=1}^{k}\left(-c\sum_{i=1}^{n'} p'_i \log_m p'_i\right)$$

Now since

$$n = \sum_{j=1}^{k} n'$$

and

$$p_i = \frac{p'_i}{m}$$

we have

$$\begin{aligned} E(C) &= c - \frac{c}{m}\sum_{i=1}^{n} mp_i \log_m(mp_i) \\ &= c\left(1 - \sum_{i=1}^{n} p_i \log_m(mp_i)\right) \\ &= c\left(1 - \sum_{i=1}^{n} p_i \log_m m - \sum_{i=1}^{n} p_i \log_m p_i\right) \\ &= -c\sum_{i=1}^{n} p_i \log_m p_i \end{aligned}$$

and the theorem is proved. *Q. E. D.*

B Unequal Test Outcomes

Tests normally do not have equally likely outcomes. What can we say about tests whose outcomes are not equally likely? It is possible to analyze this problem analytically if we assume

1. the distribution of probabilities of the m outcomes of each test is constant,
2. the probabilities of each outcome are all powers of $\frac{1}{k}$, and
3. all k^n events are equally likely.

Let the outcomes of the test be o_i, $i = 1, 2, \ldots m$ and denote their probabilities by $P(o_i)$. Without loss of generality assume that o_1 is the most likely outcome. Say its probability is $\frac{1}{k}$ for some k. Then the expected depth of the search tree is

$$\begin{aligned} E(k^n) &= c + \sum_{i=1}^{m} P(o_i) E(k^n P(o_i)) \\ &= c + \sum_{i=1}^{m} P(o_i) E(k^{n+\log_k P(o_i)}) \end{aligned}$$

If we rewrite this equation it can be put in a form in which it is readily recognizable as a difference equation. Let

$$\Delta_i = -\log_k P(o_i).$$

and

$$a_n = E(k^n).$$

Then

$$a_n = c + \sum_{i=1}^{m} P(o_i) a_{n-\Delta_i}$$

or

$$a_n - \sum_{i=1}^{m} P(o_i) a_{n-\Delta_i} = c. \qquad (1)$$

Ignoring initial conditions for now, we can try to find a solution to the difference equation by trying a general form and solving for the undetermined coefficients. A good rule of thumb is to try an expression that looks like the right hand side when the equation is written in the form of Equation 1. Trying a constant solution b_0 does not work because then the left side sums to zero. If we try a linear solution $b_1 n$ we get by substituting into Equation 1

$$b_1 = \frac{c}{\sum_{i=1}^{m} P(o_i)\Delta_i},$$

and so we have found a particular solution to the difference equation. The difference between any two solutions to a nonhomogeneous equation with constant coefficients is a solution to the homogeneous equation

$$a_n - \sum_{i=1}^{m} P(o_i) a_{n-\Delta_i} = 0.$$

Since a_n is a just a weighted average of previous solutions it is easy to see that the solutions to the homogeneous equation do not grow with n. Therefore any solution to the nonhomogeneous equation is the particular solution we found above, to within a factor that does not grow with n.

Replacing a_n by the original notation we have

$$E(k^n) = b_1 n$$

or letting $n_t = k^n$ and substituting for b_1

$$E(n_t) = \frac{c \log_k n_t}{-\sum_{i=1}^{m} P(o_i) \log_k P(o_i)}.$$

Here again an entropy-type factor is introduced.

C Different Tests

What if different tests have different distributions of outcomes? Suppose there are T different tests t_i, each of which has k_i equally likely outcomes, for $i = 1, 2, \ldots T$. Suppose each test is used with probability $P(t_i)$ and has cost c. Then an idea of how to obtain a combined estimate of expected cost can be obtained by imagining the successive application of the tests in proportion to their relative probabilities. Consider it as a combined test. The probability of each outcome will be

$$P(o) = \prod_{i=1}^{T} (\frac{1}{k_i})^{\beta P(t_i)}$$

where

$$\beta = \mathrm{lcm}(\frac{1}{P(t_i)})$$

assuming the $P(t_i)$'s are rational. This is just as if one test with

$$\prod_{i=1}^{T} (k_i)^{P(t_i)}$$

outcomes was repeated β times. So we see that the geometric mean of the numbers of outcomes of each test is the effective number of outcomes.

References

[Bolles and Cain 1982] Robert C. Bolles and Ronald A. Cain, "Recognizing and Locating Partially Visible Parts: The Local-Feature-Focus Method", *International Journal of Robotics Research*, 1(3):57–82, 1982.

[Bolles and Horaud 1986] Robert C. Bolles and Patrice Horaud, "3DPO: A Three-Dimensional Part Orientation System", *The International Journal of Robotics Research*, 5(3):3–26, 1986.

[Burns and Kitchen 1987] J. Brian Burns and Leslie L. Kitchen, "Recognition in 2D Images of 3D Objects from Large Model Bases Using Prediction Hierarchies", Technical Report, University of Massachusetts, Amherst, June 1987.

[Cooper and Hollbach 1987] Paul R. Cooper and Susan C. Hollbach, "Parallel Recognition of Objects Comprised of Pure Structure", *Proceedings of the Image Understanding Workshop*, pages 381–391, 1987.

[Freeman and Chakravarty 1980] H. Freeman and I. Chakravarty, "The Use of Characteristic Views in the Recognition of Three-Dimensional Objects", *Pattern Recognition in Practice*, pages 277–288, North-Holland, 1980.

[Fu 1968] K. S. Fu, *Sequential Methods in Pattern Recognition and Machine Learning*, Academic Press, 1968.

[Goad 1987] Chris Goad, "Special Purpose Automatic Programming for 3D Model-Based Vision", *Readings in Computer Vision*, pages 371–381, 1987.

[Jones 1979] D. S. Jones, *Elementary Information Theory*, Oxford University Press, 1979.

[Koenderink and van Doorn 1976] J. J. Koenderink and A. J. van Doorn, "The Singularities of the Visual Mapping", *Biological Cybernetics*, 24:51–59, 1976.

[Shannon 1948] C. E. Shannon, "A Mathematical Theory of Communication", *The Bell System Technical Journal*, 27(3):379–423, 1948.

[Slagle and Lee 1971] James R. Slagle and Richard C. T. Lee, "Application of Game Tree Searching Techniques to Sequential Pattern Recognition", *Communications of the ACM*, 14(2), 1971.

[Turney *et al.* 1985] Jerry L. Turney, Trevor N. Mudge and Richard A. Volz, "Recognizing Partially Occluded Parts", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-7(4):410–421, 1985.

[Wang and Suen 1984] Qing Ren Wang and Ching Y. Suen, "Analysis and Design of a Decision Tree Based on Entropy Reduction and Its Application to Large Character Set Recognition", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-6:406–417, 1984.

MODELING SENSORS AND APPLYING SENSOR MODEL TO AUTOMATIC GENERATION OF OBJECT RECOGNITION PROGRAM

Katsushi Ikeuchi and Takeo Kanade

Department of Computer Science and Robotics Institute
Carnegie-Mellon University
Pittsburgh, PA 15213

Abstract

A model-based vision system requires models to predict object appearances. How an object appears in the image is a result of interaction between the object properties and the sensor characteristics. Thus, in model-based vision, we ought to model the sensor as well as modeling the object. In the past, however, the sensor model was not used in model-based vision or, at least, was contained in the object model implicitly.

This paper presents a framework between an object model and the object's appearances. We consider two aspects of sensor characteristics: sensor detectability and sensor reliability. Sensor detectability specifies what kinds of features can be detected and in what condition the features are detected; sensor reliability is a confidence for the detected features. We define the configuration space to represent sensor characteristics. We propose a representation method for sensor detectability and reliability in the configuration space. Finally, we investigate how to apply the sensor model to a model-based vision system, in particular, automatic generation of an object recognition program from a given model.

1. INTRODUCTION

A model-based vision system requires models to predict object appearances. Various researchers propose many kinds of object models, ranging from generic models such as generalized cylinders[1, 2], extended Gaussian images[3, 4], and super quadric models[5], to specific models such as aspect models[6, 7, 8], region-relation models[9], and smooth local symmetry[10].

The object appearances, however, are determined by a *product* of an object model with a sensor model. As shown in Figure 1, the same object model in the same attitude can create different appearances and features when seen by different sensors. Edge-based binocular stereo reliably detects depth at edges perpendicular to the epipolar lines. Photometric stereo or a light-stripe range finder detects surface orientation and depth of surfaces which are illuminated and visible both by the light source and by the cameras.

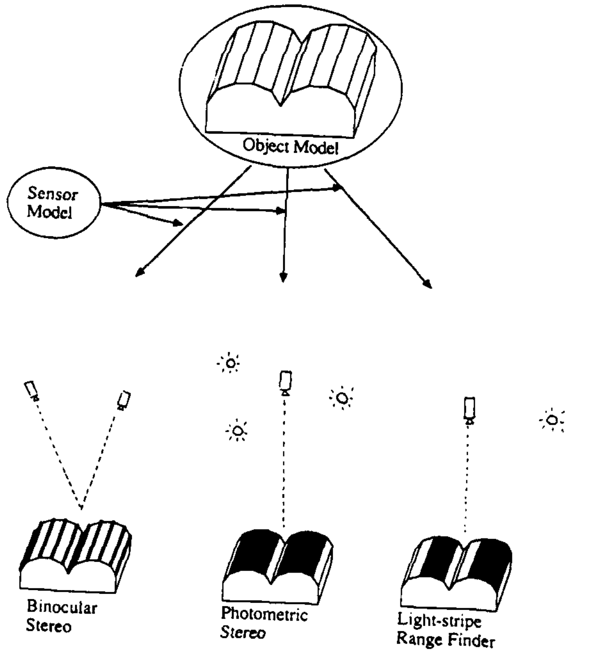


Figure 1: Object-appearances.

Thus, in model based vision, it is insufficient to consider only an object model; it is essential to exploit a sensor model

This research was sponsored by the Defense Advanced Research Projects Agency, DOD, through ARPA Order No. 4976, monitored by the Air Force Avionics Laboratory under contract F33615-84-K-1520, and under The Analytic Sciences Corporation sub constract 87123 modification No.1. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or of the U.S. Government.

as well. On the other hand, modeling sensors for model-based vision has attracted little attention; quite often, researchers who are familiar with the sensors they use tended to construct object appearances by implicitly incorporating their sensor behavior. This paper, in contrast, explores a general framework for explicitly incorporating sensor models which govern the relationship between object models and object appearances.

A sensor model must be able to specify two important characteristics: sensor detectability and sensor reliability. The sensor detectability specifies what kind of features can be detected in what condition. The sensor reliability is a measure of confidence for the detected features. This paper presents a method for modeling sensors with sensor detectability and sensor reliability, and how to use them in model-based vision. We define the configuration space to represent sensor characteristics. Next, we consider two aspects of sensor characteristics: sensor detectability and sensor reliability. We propose a representation method for sensor detectability, and examine how to use the representation for generating object appearances. Sensor reliability analysis consists of determining uncertainty in sensory measurement and analyzing uncertainty propagation from sensory measurements to geometric features. Finally, we investigate the way to use these sensor models for model-based vision, in particular, our on-going project, automatic generation of object recognition programs.

2. SENSORS IN MODEL-BASED VISION

Different types of sensors are used in model-based vision. For our purpose, "sensors" are transducers which transform "object features" into "image features". For example, an edge detector detects edges of an object as lines in an image. Photometric stereo measures surface orientations of surface patches of an object. There are both passive and active sensors. Binocular stereo is passive, while a light-stripe range finder is an active sensor using actively controlled lighting. Table 1 gives a summary of various sensors in terms of what object features are detected in what forms.

Table 1 Summary of Sensors

Sensor	Vertex	Edge	Face	active/passive
Edge Detector[11, 12, 13]	-	line	-	passive
Shape-from-shading[14, 15]	-	-	region	passive
Synthetic Aperture Radar[16]	point	point/line	point	active
Time-of-Flight Range Finder[17, 18]	-	-	region	active
Light-stripe Range Finder[30]	-	-	region	active
Binocular Stereo[19, 20]	-	line	-	passive
Trinocular Stereo[21]	-	line	-	passive
Photometric Stereo[22, 23]	-	-	region	active
Polarimetric light detector[24]	-	-	point	active

In addition to qualitative descriptions of a sensor, a sensor model must describe two characteristics quantitatively: *detectability* and *reliability*. Detectability specifies what kind of features can be detected in what conditions. Reliability specifies the expected uncertainty in sensory measurement and geometric features derived from measurement. Since these two characteristics depend on how the sensor is located relative to an object feature, we will first define a feature configuration space to represent the spatial relationship between the sensor and the feature. Then, we will investigate the way to specify detectability and reliability over the space.

3. FEATURE CONFIGURATION SPACE

Whether and how reliably a sensor detects an object feature depend on various factors: distance to a feature, attitude of a feature, reflectivity of a feature, transparency of air, ambient lighting, and so forth. In most model-based vision problems, the attitude of a feature, that is, angular freedom in the relationship between a feature and a sensor, affects sensor characteristics most. In order to specify this freedom explicitly, we attach a coordinate system to an object feature and consider the relationship between the sensor coordinate system and the feature coordinate system. For example, for a face feature, we define a coordinate system so that the z axis of the feature coordinate system agrees with the surface normal and the x-y axes lie on the face, but are defined arbitrarily otherwise. For other features, we can define feature coordinates appropriately. See Appendix I for more details.

Since angular relationships between the two coordinate systems are relative, for the sake of convenience we fix the sensor coordinate system and discuss how to specify feature coordinates with respect to it. The angular relation from the sensor coordinate system to a feature coordinate system can be specified by three degrees of freedoms: two degrees of freedom in the direction of the z axis of a feature coordinate system, and one degree of freedom in the rotation about the z axis. See Figure 2 (a).

Since we will consider the angular relationship, we can translate the feature coordinates so that the two coordinate systems share the origin. Then, we will define a sphere whose origin is located at the origin of the sensor coordinate system, and whose north pole is on the z axis of the sensor coordinate. We will specify a feature coordinate as a point in the sphere. Referring to Figure 2 (b), the north pole of the sphere is made to correspond to the case when the feature coordinates are aligned completely with the sensor coordinates. For other feature coordinates, the direction from the sphere center to the point coincides with the z axis of the feature. The distance from the spherical surface to the point is determined by the angle of rotation (modulo 360°) around the z axis from the coordinate on the spherical surface. A point on the spherical surface represents a feature coordinate obtained by rotating the sensor coordinate around the axis perpendicular to plane given by the sphere center, the spherical point, and the north pole. We will refer to this sphere as the feature configuration space.[1] Figure 2(c) shows various feature coordinates corresponding to spherical points, while Figure 2(d) shows those corresponding to points on a radical axis.

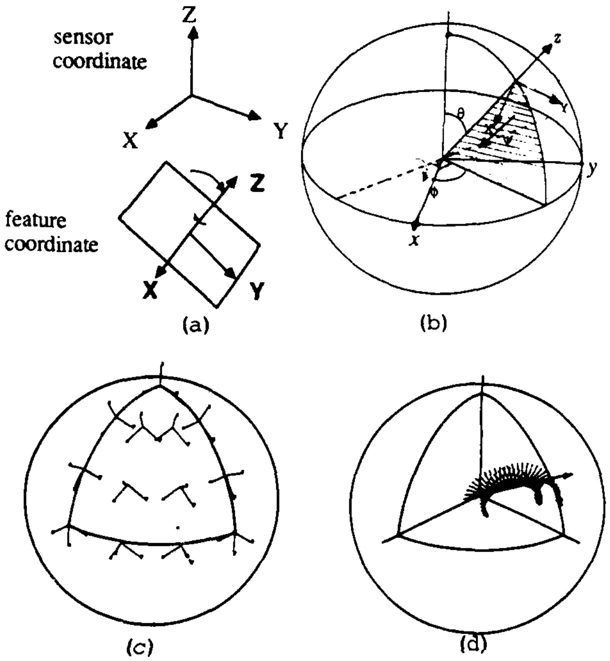


Figure 2: Feature configuration space: (a) A feature coordinate with respect to the sensor coordinate; (b) A feature configuration space; (c) Feature coordinates on the spherical surface; (d) Feature coordinates along a radial axis.

4. DETECTION CONSTRAINTS and ASPECTS

In the previous section, we have defined the way to represent the relationship between sensor coordinates and feature coordinates. In this section, we will develop a constraint to determine whether a feature can be detected at each point of the configuration space.

4.1. Detection Constraints

Each sensor has two components: illuminators and detectors. For example, both a time-of-flight range finder and a light-stripe range finder have one light source and one TV camera. Binocular stereo has one light source (without light sources you cannot observe anything) and two TV cameras; photometric stereo has three light sources and one TV camera. Table 2 summarizes the number of illuminators and detectors of each sensor.

Table 2 Illuminator and Detector

Sensor	*Number of illuminators*	*Number of detectors*
Edge Detector	1	1
Shape-from-shading	1	1
SAR	1	1
Time-of-Flight Range Finder	1	1
Light-stripe Range Finder	1	1
Binocular Stereo	1	2
Trinocular Stereo	1	3
Photometric Stereo	3	1
Polarimetric light detector	n	1

One illuminator only illuminates one part of an object; one detector only observes one part of the object. Each sensor, which consists of illuminators and detectors, only detects one part of the object. In order for a feature to be detected by a sensor, the feature must satisfy certain conditions on being illuminated by its illuminators and being visible from its detectors.

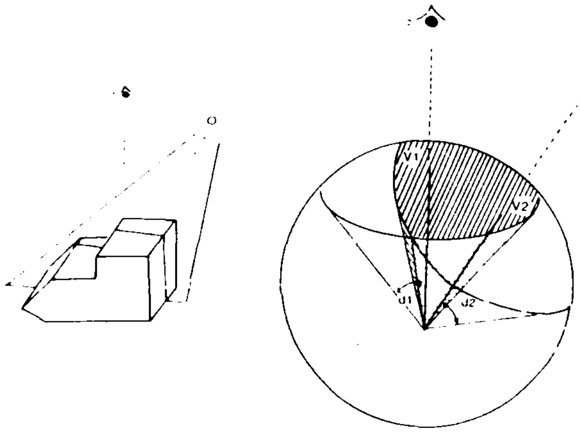
Once we define a local coordinate system on a feature, we can compute configuration of a feature in which it is illuminated by each illuminator, and configurations in which it is visible by each detector. In the following discussion, we will consider both illuminators and detectors as generalized sources (G-sources). Each G-source has two properties: the illumination direction and the illuminated configurations. In the source case, these two terminologies are the same as the nominal meanings. In the case of detectors, the illumination direction corresponds to the line of sight of the detector, and the illuminated configurations correspond to the visible configurations from the detector.

G-source illumination direction can be represented in the feature configuration space as a radial line from the sphere center. G-source illuminated configurations can be specified as a volume in the configuration space. Finally, we can obtain the constraints in which the feature is detectable by the sensor with AND operations on illumination directions and illuminated configurations of all component G-sources of the sensor.

Figure 3(b) shows an example analysis of a face feature for a light-stripe range finder in Figure 3(a). A light-stripe range finder has two G-sources (a TV camera and a light source): the direction denoted by $V1$ indicates the line of sight of the TV camera; $V2$ indicates the illumination direction of the light source. The illuminated configurations of a face from the light source are determined by the z axis (ie, its surface normal) of a face feature coordinate, and are not dependent on its rotation. Therefore, illuminated configurations of a feature

[1] This representation will not create discontinuities around the north pole as opposed to the case in which Euler angles from the sensor coordinate frame to the feature coordinate frame are used to specify spherical points; this representation will instead create discontinuities at the center of the sphere and at the south pole. However, this is advantageous because we mostly use the area around the north pole to discuss detectability and reliability.

form a spherical cone whose axis is *V*2 and whose apex angle is *d*2. Similarly, the configurations of a feature visible from the TV camera form a spherical cone whose center direction is *V*1 and whose apex angle is *d*1. Since a light-stripe range finder detects the faces which are illuminated from the source and visible from the TV camera, the detectable configurations (necessary condition) are the intersection of the two cones. Thus, the resulting detection constraints in Figure 3(b) contain three constraint components; two illumination directions and one volume of detectable configurations.



(a) (b)

Figure 3: Detection constraints of a light-stripe range finder: (a) Our hypothetical light-stripe range finder; (b) Constraints in the feature configuration space; Note that there are two kinds of *constraints; illuminated configurations* and illumination directions.

Similarly we can analyze the detection constraints for various sensors in Table 1. The results of the analysis are summarized in Figure 4. The right column represents each sensors detection constraints in the feature configuration space. The center column represents our formal language to specify the detection constraints to our geometric modeler to be used for generating object appearances. The definition of the language is found in appendix II.

4.2. Use of Detection Constraints

To predict object appearances, we apply the constraints to each feature of the object. Each feature is detectable by the sensor if it satisfies the following two conditions:

1. None of the illumination directions are occluded by any other parts of the object
2. The detectable configurations contain the configuration of the feature.

To check these conditions we use the constraints with a geometric modeler. We rotate the object into a certain attitude to be examined, and then see whether its features satisfy the previous constraints. Figure 5 illustrates this

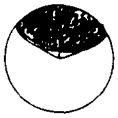
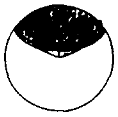
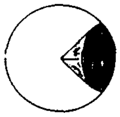
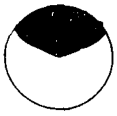
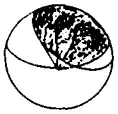
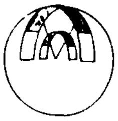
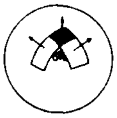
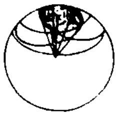
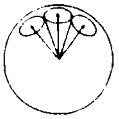
Sensor	Constraints *in the formal definition*	Constraints *in the sensor space*
Edge Detector	(AND (NS edge V d) (NS edge V d)) = (NS edge V d)	
Shape-from-shading	(AND (NS face V d) (NS face V d)) = (NS face V d)	
SAR	(OR (NS face V d) (NS edge V d) (NS vertex V d)) (needs postprocess)	
Time-of-Flight Range Finder	(AND (NS face V d) (NS face V d)) = (NS face V d)	
Light-strip Range Finder	(AND (NS face V1 d) (NS face V2 d))	
Binocular Stereo	(AND (NS edge V1 d1) (DS edge V2 d2 VE de) (DS edge V3 d3 VE de))	
Trinocular Stereo	(AND (NS edge V1 d1) (DS edge V2 d2 VE de) (DS edge V3 d2 VE de) (DS edge V4 d2 VE de))	
Photometric Stereo	(AND (NS face V d1) (NS face V1 d2) (NS face V2 d2) (NS face V3 d2))	
Polarimetric Light Detector	(OR (AND (NS face V d) (NS face V1 d)) (AND (NS face V d) (NS face V2 d))) where V V = cos 2d	

Figure 4: Detection constraints for various sensors.

process of predicting object appearances for a light-stripe range finder. Suppose an object is placed like Figure 5 (a). Figure 5 (b) shows the detection constraints on a face for a light-stripe range finder. We will put this configuration space on each candidate face to examine whether the face is detectable. See Figure 5(c). This amounts to checking the following conditions:

1. The light source direction is not occluded by other faces.
2. The line of sight of the TV camera is not occluded by other faces.
3. The local coordinate of an face, defined by the surface normal (z axis) and the tangential plan (x-y axis), is contained in the detectable configurations.

Figure 5(d) shows the result of this operation. The shaded areas indicate those which satisfy the conditions and thus are detectable by the light-stripe range finder.

4.3. Aspect

By using object appearances by the detection constraints, we can generate aspects defined by a particular sensor. Aspects are originally defined as topologically equivalent classes with respect to the object features[6]. Since each sensor has particular features to be detected, we have to modify the definition of aspects according to features detected by a sensor. We will consider aspects given from appearances by a light-stripe range finder.

We can define aspects for a light-stripe range finder by those faces detectable in each aspect. Suppose we have n faces, $S_1, S_2, \ldots, S_n$. Let the variable X_i denotes the condition whether or not the face S_i is detected, that is

$$X_i = \begin{cases} 1 & \text{face } S_i \text{ is detectable;} \\ 0 & \text{otherwise.} \end{cases}$$

An n-tuple $(X_1, X_2, \ldots, X_n)$ represents a label of an object appearance in terms of face detectability. This label will be referred to as a *shape label*, and we can characterize each object attitude with this label. The set of contiguous object attitudes that have the same *shape label* forms an *aspect* in this example.

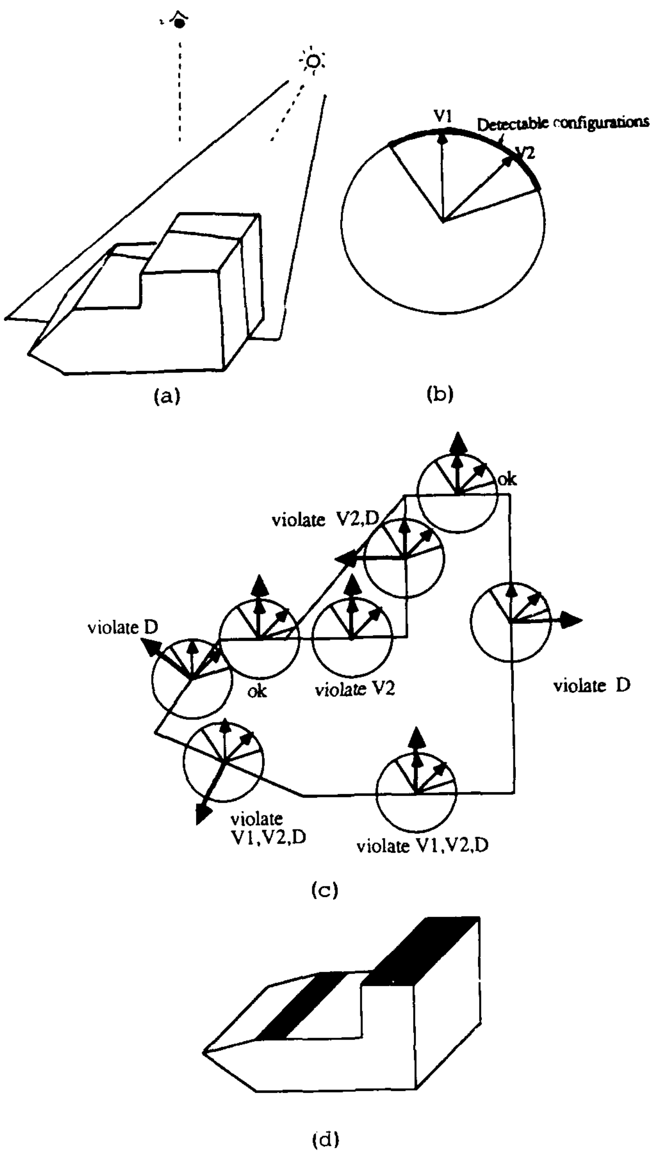


Figure 5: How to use the constraints: (a) A light-stripe range finder; (b) The detection constraints; (c) Applying the detection constraints to object features; (d) Detectable faces determined.

5. DETECTABILITY DISTRIBUTION AND ASPECT TRANSIT

5.1. Detectability Distribution

The feature detection constraint gives the upper bound of the detectable configurations in the space. In some cases, however even though an object feature satisfies the detection constraints, it may be undetected due to noise. We define the detectability distribution such that a feature in the detectable configurations is actually detected. The probability is usually high in the central part and low in the peripheral part of the detectable configurations.

Since all sensors in Table 1 detect features based on a brightness value corresponding to a feature, the detectability distribution depends on a brightness value which is detected and converted to sensor features. However, there are two *types of sensors in terms of the conversion method; intensity* based sensors and position based sensors. The intensity based sensor measures the brightness value and converts it to sensor features directly. The position based sensor measures the brightness value and gets positional information of the bright spot if the brightness value is greater than some threshold. The position based sensor then converts the positional information to sensor features. Table 3 shows a classification of sensors based on this difference.

Since the detectability distribution depends on sensing methods, we will develop the detectability distribution for a light-stripe range finder as a representative case of the position based sensor, and the distributions for photometric stereo as a representative case of the intensity based sensor.

Table 3 Measurement method

sensor	sensing method
Edge Detector	intensity
Shape-from-shading	intensity
SAR	intensity
Time-of-Flight Range Finder	position
Light-stripe Range Finder	position
Binocular Stereo	position
Trinocular Stereo	position
Photometric Stereo	intensity
Polarimetric light detector	intensity

5.1.1. Detectability distribution of light-stripe range finder

We will consider the detectability distribution of a hypothetical li:ht-stripe range finder. A light-stripe range finder projects a plane of light onto the scene and determines the position of a surface patch from the slit image. The detectability depends on whether the brightness of the slit image is bright enough to be detected, say brighter than a threshold I_0. Assuming a Lambertian surface, the brightness of the slit image is given by $I_s\mathbf{N}\cdot\mathbf{S}$ where N is the surface normal, S is the light source direction, and I_s is the light source brightness. If we assume an additive zero-mean Gaussian noise of brightness with power σ^2, the resultant brightness distribution of a slit will be

$$p(I) = \frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{(I-I_s\mathbf{N}\cdot\mathbf{S})^2}{2\sigma^2}}.$$

Thus, the probability distribution of feature detectability of our hypothetical range finder can be described as

$$P_d = Prob(I \geq I_0)$$
$$= \int_{I_0-I_s\mathbf{N}\cdot\mathbf{S}}^{+\infty} \frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{I^2}{2\sigma^2}}dI$$
$$= \Phi(\frac{I_0-I_s\mathbf{N}\cdot\mathbf{S}}{\sigma})$$

As shown in Figure 6, this probability decreases as the incident angle of the light-stripe increases, and near the boundary of the illuminated configuration of the light source, the probability approaches 0.

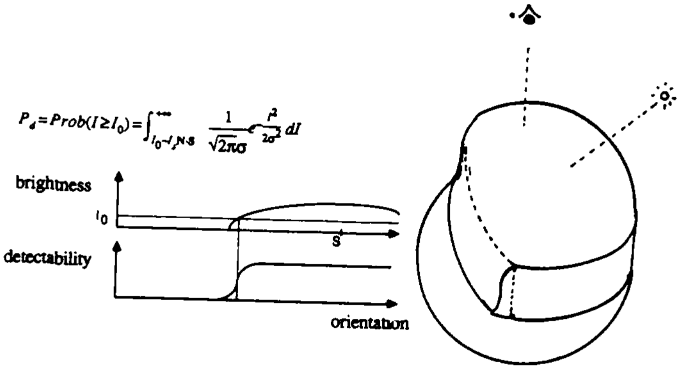

$P_d=Prob(I\geq I_0)=\int_{I_0-I_sN\cdot S}^{+\infty}\frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{I^2}{2\sigma^2}}dI$
brightness
I_0
S
detectability
orientation

Figure 6: Detectability distribution for a light-stripe range finder

5.1.2. Detectability distribution of photometric stereo

An intensity based sensor can be modeled as

$$\mathbf{y}=\mathbf{G}(\mathbf{x})$$

where **x** is the input brightness, **y** is the output feature values, and **G** is the conversion function. Suppose $\mathbf{X}^*$ is the definition area of the function **G**; ie, the intensity based sensor outputs a feature value y from any $\mathbf{x}_i$ if $\mathbf{x}_i \in \mathbf{X}^*$. Then, the detectability distribution can be determined as the probability that the input brightness, $\mathbf{x}+\delta\mathbf{x}$, disturbed by $\delta\mathbf{x}$, is still contained in the definition area, $\mathbf{X}^*$.

Photometric stereo determines the surface orientation from three images taken from the same position under different lighting dir··tions.

$$I_1=\mathbf{S}_1\bullet\mathbf{N}$$
$$I_2=\mathbf{S}_2\bullet\mathbf{N}$$
$$I_3=\mathbf{S}_3\bullet\mathbf{N},$$

where I_i,$\mathbf{S}_i$,N are the brightness value under light source i, the i th light source direction vector, and the surface normal vector, respectively. Thus, expressing the brightness as a vector, **I**, and the light source as a matrix, **S**,

$$\mathbf{I}=\mathbf{S}\,\mathbf{N}.$$

Applying $\mathbf{S}^{-1}$ to both sides, we obtain an explicit expression of **N**,

$$\mathbf{N}=\mathbf{S}^{-1}\,\mathbf{I}=\mathbf{A}\,\mathbf{I},$$

where $\mathbf{A}=\mathbf{S}^{-1}$. This is the basic idea of photometric stereo[22].

The definition area of **I** is determined as an inverse area of the valid surface orientations. Namely, since **N** is a unit vector, a valid surface orientation **N** satisfies $\mathbf{N}^t\mathbf{N}=1$. Once we insert the relationship between surface orientation and brightness triple, we obtain $(\mathbf{A}\,\mathbf{I})^t\mathbf{A}\,\mathbf{I}=1$, which gives the definition area of **I**.

Working photometric stereo has, however, three modifications[23] to the original theory.

1. Brightness values are normalized $\mathbf{I}/|\mathbf{I}|$ so that we can cancel the albedo effect.

2. Threshold operations are applied to brightness values so that we can eliminate shadow regions.

3. A is determined from calibration and stored in a lookup table rather than calculated from the ideal case.

We will obtain the detectability distribution of the photometric stereo according to these modifications. Assume a brightness value moves from i_1 to $i_1+\delta i_1$ due to TV camera's error. The normalization gives $i'_1+\delta i'_1=(i_1+\delta e_1)/(i_1+\delta i_1+i_2+i_3)$. However, the normalized intensity $(i'_1+\delta i'_1,i'_2,i'_3)$ exists on the same plane $i'_1+\delta i'_1+i'_2+i'_3=1$. Since a continuous area on the plane is the definition area for photometric stereo and the new triple $i'_1+\delta i'_1,i_2,i_3$ is still contained in the definition area on the plane, we can obtain the solution from the new triple.

That is, we will always succeed in obtaining the feature values, ie. we will have a unit detectability distribution for the light source 1. (Though of course the resultant value may be less reliable as will be discussed in the reliability section.) The same discussion is applicable to light source 2 and light source 3. This analysis reveals that normalization makes the detectability a unit value, and thus, helps to detect features in a stable manner.

We use the threshold operations to detect shadows. This operation also affects the detectability distribution. Namely, in the case that all three brightness values are greater than a threshold value, we can determine the surface orientation. This effect can be modeled in the same way as the light stripe range finder. Thus,

$$P_d=\Phi\left(\frac{I_0-I_{s1}\mathbf{N}\cdot\mathbf{S1}}{\sigma}\right)\Phi\left(\frac{I_0-I_{s2}\mathbf{N}\cdot\mathbf{S2}}{\sigma}\right)\Phi\left(\frac{I_0-I_{s3}\mathbf{N}\cdot\mathbf{S3}}{\sigma}\right),$$

gives the detectability distribution over the detectable configurations, where I_{si},**Si** is the light source brightness, the light source direction of the *i* illuminator, respectively.

5.2. Transition of Aspect

The continuous change of detectability causes the continuous aspect transition and the aspect boundaries to become blurred. In order to characterize an aspect boundary, we can define the distance between two aspects across the boundary by the Hamming distance between their corresponding shape labels $\{x_1,x_2,...,x_i,...,x_n\}$, where $x_i=1$ if face i is detected and $x_i=0$ otherwise. Thus, the distance of two aspects is the number of faces which switch between detected and nondetected states across the aspect boundary.

Consider an aspect boundary between aspect A and B whose Hamming distance is one, that is, aspect A and B differ in detection of only one face F_i. Suppose the detectability of face i is $P_d(F_i)$. Then, near the aspect boundary, the aspect A may be observed incorrectly as aspect B with probability $1-P_d(F_i)$. A similar false observation will also occur for aspect B.

If the Hamming distance of aspects A and B is more than one across a boundary, then erroneous intermediate aspects, which are neither A nor B, can occur near the boundary. This can be easily seen by considering an example where aspect A has $\{x_i,x_j\}=\{10\}$ and aspect B has $\{x_i,x_j\}=\{01\}$ as shape labels, respectively. Then, we will observe object appearances belonging to four aspects near the boundary: aspects $\{11\}$ and $\{00\}$ in addition to aspects A and B. For example, the probability of observing aspect $\{11\}$, instead of aspect A, is $P_d(F_i)P_d(F_j)$. This consideration must be taken into account when grouping and classifying aspects by an interpretation tree.

6. RELIABILITY OF SENSORS

Once a sensor feature is detected, then the next question is how reliable the sensor feature is. This section discusses two issues of uncertainty in feature values. The first issue is uncertainty in sensory measurement; any sensory measurement detected by a sensor always contains measurement uncertainty. To determine the bound of the uncertainty is important for model based vision. For example, suppose there is a sensor feature for which the geometric model takes two nominal values, 100 and 90, for two distinct situations. If a sensor has an uncertainty range of plus/minus 1 for the sensor feature, we can use the feature from that sensor as one of reliable discriminators in the recognition stage. On the other hand, if a sensor has an uncertainty range of plus/minus 20, we cannot use the feature from that sensor.

The second issue is propagation of uncertainty from sensor features to geometric features, hence the resulting uncertainty of those geometric features. In some cases, a detected sensor feature from a sensor is used directly as a feature; in most cases, however, geometric features are derived from sensor features. Thus, it is necessary to determine the uncertainty propagation mechanism for determining resulting uncertainty in geometric features.

6.1. Uncertainty in Sensory Measurement

Uncertainty in sensor measurement comes from various reasons; variance in brightness values, variance in light source direction, various digitization mechanisms. However, the major uncertainty of an intensity-based sensor comes from brightness variance, and the major uncertainty of a position-based sensor comes from variance in light source as show n in Table 4.

Table 4 Main factor of unreliability

Sensor	Type	Factor
Edge Detector	Int	brightness variance
Shape-from-shading	Int	brightness variance
SAR	Int	flight direction
Time-of-Flight Range Finder	Pos	mirror direction
Light-stripe Range Finder	Pos	mirror direction
Binocular Stereo	Pos	camera directions
Trinocular Stereo	Pos	camera directions
Photometric Stereo	Int	brightness variance
Polarimetric light detector	Int	polarimetric variance

Since uncertainty in sensory measurement depends on the sensor, we will analyze the light-stripe range finder as a representative position-based sensor and photometric stereo as a representative intensity-based sensor.

6.1.1. Light-stripe range finder

We will consider a depth measurement by a hypothetical light-stripe range finder. Let us assume that the main source of uncertainty in depth measurement comes from the ambiguity of the slit position on a surface due to the width of the light beam and angular errors in setting the light directions. The uncertainty model can be obtained analytically.

As shown in Figure 7 (a), let us denote the angular uncertainty of the light stripe by $\delta\theta$. The light is intercepted by an object surface, creating a slit pattern on it. The angular uncertainty $\delta\theta$ of the light direction results in uncertainty δy in the position on the surface:

$$\delta y=\frac{r\delta\theta,}{\cos\alpha}$$

where r is the distance of the surface from the light source, and α is the angle between the light direction **S** and the surface normal **N**. This positional uncertainty on the surface is observed as the slit position uncertainty (or "slit width") δi in the camera image. If β is the angle between the surface normal **N** and the viewer direction **V**, then

$$\delta i=(\cos\beta)\delta y,$$

Finally, this uncertainty is transferred into the uncertainty in depth measurement by triangulation. For simplicity, if we assume orthographic projection for the camera, the uncertainty in the image δi creates uncertainty in depth δz,

$$\delta z=\frac{\delta i}{\tan\gamma},$$

where γ is the angle between **V** and **S**.

In total, by representing the angles α, β, and γ in terms of **V**, **N**, and **S**, we obtain

$$\delta z=\frac{\cos\beta}{\cos\alpha\tan\gamma}r\delta\theta=\frac{(\mathbf{N}\cdot\mathbf{V})(\mathbf{S}\cdot\mathbf{V})}{(\mathbf{N}\cdot\mathbf{S})\sqrt{1-\mathbf{S}\cdot\mathbf{V}}}r\delta\theta.$$

Since r is roughly constant, the uncertainty distribution of this light-stripe range finder over the detectable configurations is governed by the factor $\frac{(\mathbf{N}\cdot\mathbf{V})(\mathbf{S}\cdot\mathbf{V})}{(\mathbf{N}\cdot\mathbf{S})\sqrt{1-\mathbf{S}\cdot\mathbf{V}}}$.

Figure 7 (b) plots this function. The left sphere represent the feature configuration space. The detectable configurations are enclosed by two great circles. Since the light-stripe range finder is independent on the rotation of the z axis of the feature coordinate system, we only plot uncertainty for the configurations on the spherical surface. The detectable configurations are projected onto the tangential plane at the north pole. The right diagram represents the distribution of uncertainty over the plane. The larger the angle between the surface normal and the illuminator direction is, the larger the uncertainty we have.

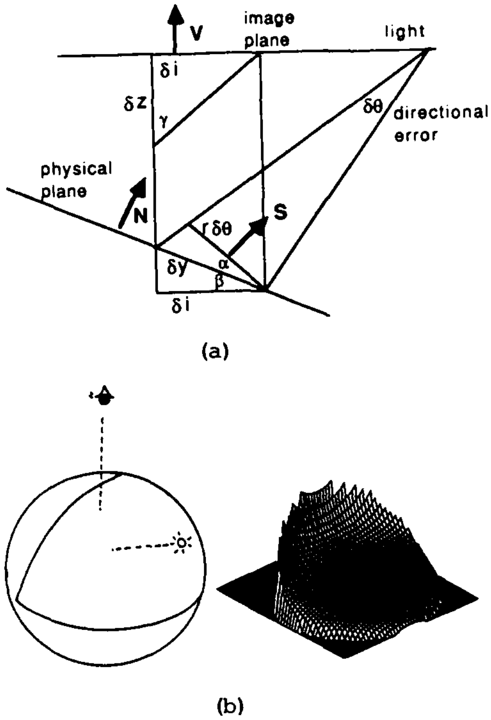


Figure 7: Predicted uncertainty in depth measurement by a light-stripe range finder: (a) Detection mechanism; (b) Predicted uncertainty in depth measurement.

6.1.2. Photometric stereo

Let consider the uncertainty in surface orientation by photometric stereo. Our photometric stereo can be described as a two step process. First, a brightness triple **I** is converted to a normalized brightness triple **E**.

$$\mathbf{E}=\mathbf{F}(\mathbf{I}).$$

Then, the normalized brightness triple is converted to a surface orientation **N**.

$$\mathbf{N}=\mathbf{A}\mathbf{E}.$$

Thus, we can obtain the uncertainty of surface orientation

$$U=(d\mathbf{N})^t d\mathbf{N}=(d\mathbf{I})^t\mathbf{J}^t\mathbf{A}^t\mathbf{A}\mathbf{J}d\mathbf{I},$$

where **J** is the jacobian matrix of **F**.

We now examine the jacobian matrix over the detectable configurations. Figure 8 (a) shows the distribution of $\frac{\partial f_1}{\partial i_1}$ over the detectable configurations, where the detectable configurations are plotted at the tangential plane at the north pole of the feature configuration space. Although it is possible to approximate the distribution with polynomial, we assume it is constant 0.004 over the detectable configurations for simplicity. We follow the same method for the other component of the jacobian matrix.

$$J=\begin{bmatrix}0.004 & 0.002 & 0.002\\ 0.002 & 0.004 & 0.002\\ 0.002 & 0.002 & 0.004\end{bmatrix}$$

We determine $\mathbf{A}d\mathbf{E}$ from the real data, because **A** is represented as a lookup table. From our experiment, our TV camera (8 bit) has standard deviation; 3. Thus, taking two sigma, di=6. Suppose only one light source causes error at one time. Then, normalized brightness uncertainty, $\sqrt{d\mathbf{E}^t d\mathbf{E}}$=0.03. Our lookup table has 16 cells in one normalized brightness axis and the distance between the maximum normalized brightness and the minimum normalized brightness is roughly 0.8 over the detectable configurations. 0.03 uncertainty in normalized brightness corresponds to 1.5 mesh uncertainty in the lookup table.

Next, we examine the relationship between one cell distance and the surface orientation difference. Figure 5b shows the angular distance between two adjacent cells in the lookup table. By using this result and a 1.5 mesh uncertainty in the normalized brightness, the total uncertainty in surface orientation becomes 5 degrees over the detectable configurations. In Figure 5c, the horizontal broken lines indicates plus/minus 5 degrees uncertainty, while vertical thick lines indicate uncertainty directly measured from our system. Both results agree over the detectable configurations.

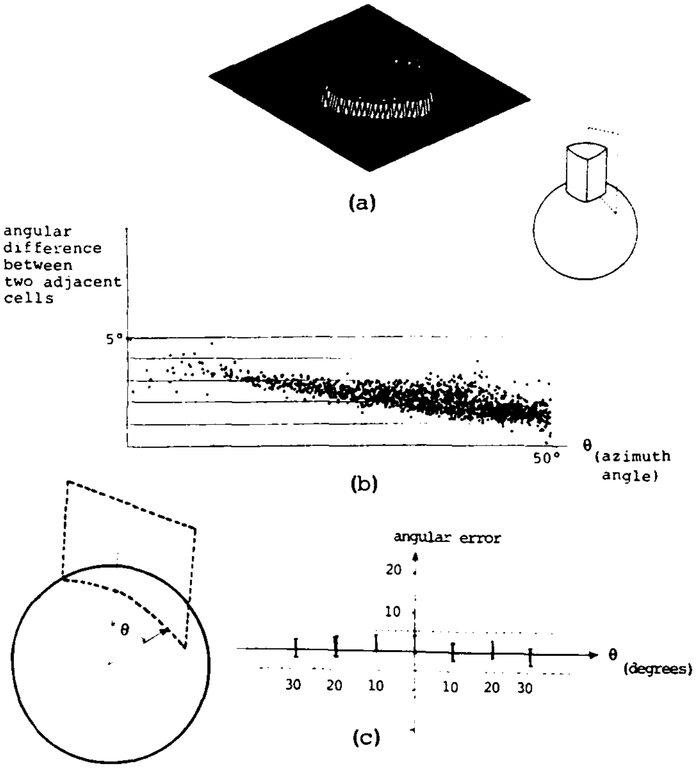


Figure 8: Uncertainty in surface orientation by photometric stereo: (a) Distribution of $\frac{\partial f_1}{\partial i_1}$ over the detectable configurations; (b) Angular difference between two adjacent cells in the lookup table; (c) Uncertainty in surface orientation over the detectable configurations.

6.2. Uncertainty in Geometric Features

Usually sensory measurements, such as depth detected by a range finder, are further converted into object features such as area and inertia of a face. This process involves grouping pixels into regions, extracting some feature values and transforming them into object features. Modeling uncertainty propagation in this process is difficult in general, but as an example of predicting the uncertainty range of a geometric feature, let us consider an area feature of a face detected as a region by our hypothetical light-stripe range finder. Figure 9(a) shows the conversion process from depth values to the area of a face. The process includes three parts: grouping pixels into a region to obtain the area in the image, computing the surface orientation of the region, and finally converting the image area into the surface area by the affine transform determined by the surface orientation. We will analyze how the uncertainty are introduced and propagated in these three parts.

Suppose that a surface under consideration has the real area A and the surface orientation β (angle between the surface normal and the viewing direction). It should create a region of size n pixels where

$$n=A\,cos\beta.$$

However, because of the imperfect detectability of the sensor, the sensor fails to find some of them, and the measured area will be different from the nominal area n. Let P_d denote the detectability for this surface which we have computed in subsection 4.2. Then, the process of measuring the area by sampling n pixels can be modeled by a binomial distribution with mean nP_d and variance $nP_d(1-P_d)$. Assuming two standard deviations, the discrepancy in area measurement will be

$$\delta n=n-(nP_d-2\sqrt{nP_d(1-P_d)})$$
$$=n(1-P_d)+2\sqrt{nP_d(1-P_d)}.$$

Another uncertainty is introduced in obtaining the surface orientation β from measured depths due to uncertainty in depth measurement δz. If we estimate the surface orientation at a pixel by differentiating depths of neighboring pixels, then the uncertainty in surface orientation will be $cos^2\beta\delta z$. However, since we have roughly n pixels in the region, the surface orientation will be averaged, reducing the uncertainty by a factor $\sqrt{n}$. Thus

$$\delta\beta=\frac{\cos^2\beta}{\sqrt{n}}\delta z$$

Finally, the estimation of area of the face, $A+\delta A$, is obtained by converting the image area into 3D surface area.

$$A+\delta A=\frac{n+\delta n}{cos(\beta+\delta\beta)}$$

Thus, assuming that $\delta\beta$ is small, we see that

$$\delta A = A(1-P_d) + 2\sqrt{\frac{AP_d(1-P_d)}{cos\beta}} + Atan\beta\delta\beta$$

$$= A(1-P_d) + \sqrt{\frac{A}{cos\beta}}(2\sqrt{P_d(1-P_d)} + \frac{sin2\beta}{2}\delta z)$$

In this way, we can predict what deviations from the nominal value of the area feature should be expected once we model the sensor and know its intrinsic detectability P_d and uncertainty in depth δz.

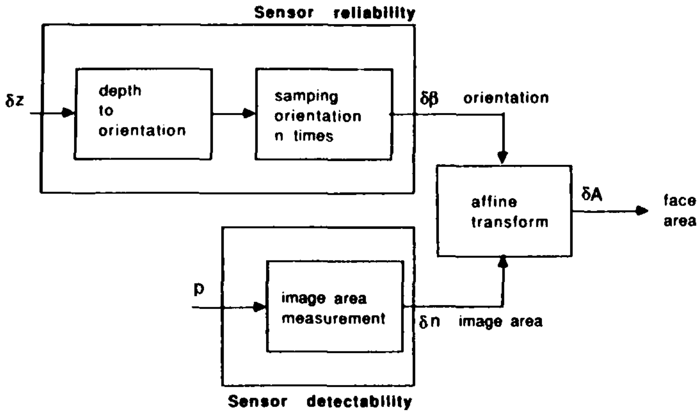


Figure 9: Conversion process from depth value values to the area of a face.

In order to examine the validity of the propagation model, we executed an experiment using our photometric stereo. We observed a face five times by our photometric stereo, where the face was inclined 30° from the direction of TV camera's axis. From the measured surface orientation and projected pixel number, we recovered the face area and face inertia by the affine transform and obtained the results shown in the column of observed in Table 5.

Our photometric stereo has plus/minus 5°,(or 0.0872 radian) uncertainty in surface orientation and observed $n = 506$ pixels from the face. This gives $\delta\beta = 0.00387$. Since we measure the face in the ideal condition, $P_d = 1$. We insert these parameters in the above equation and obtain the uncertainty in area. In order to predict the uncertainty range in inertia, we double the uncertainty range in area. These values are shown in the column of predicted in Table 5.

The result gives the consistency between the predicted uncertainty and the real uncertainty.

Table 5 Reliability of Geometric Features

Feature	Observed	Predicted
Area	0.0023	0.0022
Inertia	0.0044	0.0043

6.3. Use of Uncertainty for Tree Generation

We have to organize the structure of aspect to be convenient for applying the uncertainty formula derived in the previous section. We represents aspects and appearances in the frame system. An example of an aspect is shown in Figure 10. Each appearance frame *I0* points to several frames *IMAGE-COMP01,IMAGE-COMP02* corresponding to its visible faces. In the same way, *ASPECT1* points to several aspect component frames, *ASPECT-COM10, ASPECT-comp11*. It also points to its instance images *I0, I1*, ..., while its aspect component frame, *ASPECT-COMP10* points to its instance 2D faces *IMAGE-COMP10, IMAGE-COMP12*.

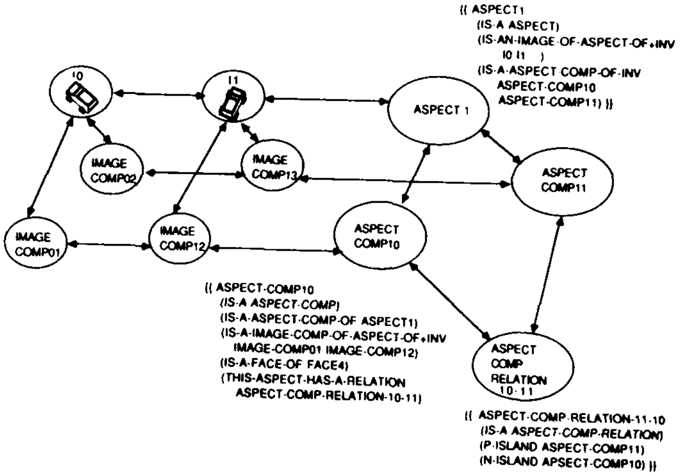

((ASPECT1
(IS-A ASPECT)
(IS-AN-IMAGE-OF-ASPECT-OF+INV
I0 I1)
(IS-A-ASPECT COMP-OF-INV
ASPECT-COMP10
ASPECT-COMP11)))
I0
I1
ASPECT 1
IMAGE COMP02
IMAGE COMP13
ASPECT COMP11
IMAGE COMP01
IMAGE COMP12
ASPECT COMP10
((ASPECT-COMP10
(IS-A ASPECT-COMP)
(IS-A-ASPECT-COMP-OF ASPECT1)
(IS-A-IMAGE-COMP-OF-ASPECT-OF+INV
IMAGE-COMP01 IMAGE-COMP12)
(IS-A-FACE-OF FACE4)
(THIS-ASPECT-HAS-A-RELATION
ASPECT-COMP-RELATION-10-11)
ASPECT COMP RELATION 10-11
((ASPECT-COMP-RELATION-11-10
(IS-A ASPECT-COMP-RELATION)
(P-ISLAND ASPECT-COMP11)
(N-ISLAND APSECT-COMP10)))

Figure 10: Frame representation of aspects: Each aspect structure consists of an aspect frame, aspect component frames, and aspect component relation frames. An aspect frame also points to its instance images *I0, I1, ...*, while its aspect component frame, *ASPECT-COMP10* points to its instance 2D faces *IMAGE-COMP01, IMAGE-COMP12.*

In order to predict the uncertainty of various feature values at each aspect, we follow the pointers between aspect component frames and appearance component frames. At each appearance component frame, since a nominal value of a feature and its configuration with respect to sensor coordinates are given, we can calculate the uncertainty of the feature value by using the formula described in the previous section. Tl :n, the uncertainty of the feature value at an aspect component is obtained as a sum of the uncertainties of the feature values over its registered image components. The predicted range will be stored in the slot of an aspect component frame. These uncertainty ranges of features will be retrieved by generation rules at compile time to generate an interpretation tree and by the execution process at run time in recognizing a scene.

7. AUTOMATIC GENERATION OF OBJECT RECOGNITION PROGRAM

Figure 11 presents an example of how to apply the sensor model to the automatic generation of an object recognition program. A geometric model as shown in Figure 11(b) is obtained from a toy wagon in Figure 11(a). From the geometric model and sensor detectability, we can generate various possible appearances as shown in Figure 11(c). We can classify and categorize various appearances into possible aspects, where each aspect shares the common detectable faces. One aspect structure is constructed at each aspect group (Figure 11(d)). Predicted ranges of uncertainty of geometric features are determined using the sensor reliability. Generation rules generate the recognition strategy automatically based on the predicted ranges of uncertainty (Figure 11(e)). Once the recognition strategy is obtained, the strategy is converted into an executable program (Figure 11(f)).

The generated program is applied to the scene as shown in Figure 12. The highest region, indicated by an arrow, is given to the program, while Figure 12 (b) shows the needle map given by our photometric stereo. The black nodes in Figure 12 (c) indicates the follow of control in the real run. The program classifies the region to the corresponding aspect as shown in Figure 12(c).

8. CONCLUDING REMARKS

This paper discussed modeling sensors and how to apply sensor modeling to automatic generation of object recognition programs. Our sensor model consists of two characteristics: sensor detectability and sensor reliability. Sensor detectability specifies under what conditions a sensor can detect a feature, while sensor reliability is a measure of confidence for the detected features over the detcetable configurations.

We have defined the configuration space which represents the relationship between sensor coordinates and object coordinates. The sensor detectability and the sensor reliability are expressed in this configuration space. Constraints in the configuration space involved in detecting features have been developed by using G-source illuminated configurations and G-source illumination directions. We have shown how to compute the sensor detectability distribution and the sensor reliability distribution for two representative sensors: a light-stripe range finder and photometric stereo.

In model based vision, expected values of various features can be computed from 3D geometric models. Those values are, however, nominal values that should be taken in ideal cases or should be sensed by ideal sensors. On the other hand, actually observed sensor data contains noise and should be used accordingly. The sensor model bridges the discrepancy between these two values by modeling the distribution of the sensed value based on the characteristics of a given sensor.

We also have analyzed the uncertainty propagation mechanism from sensory measurements to geometric features. This is important, because quite often we are interested in geometric features derived from detected measurement. Once

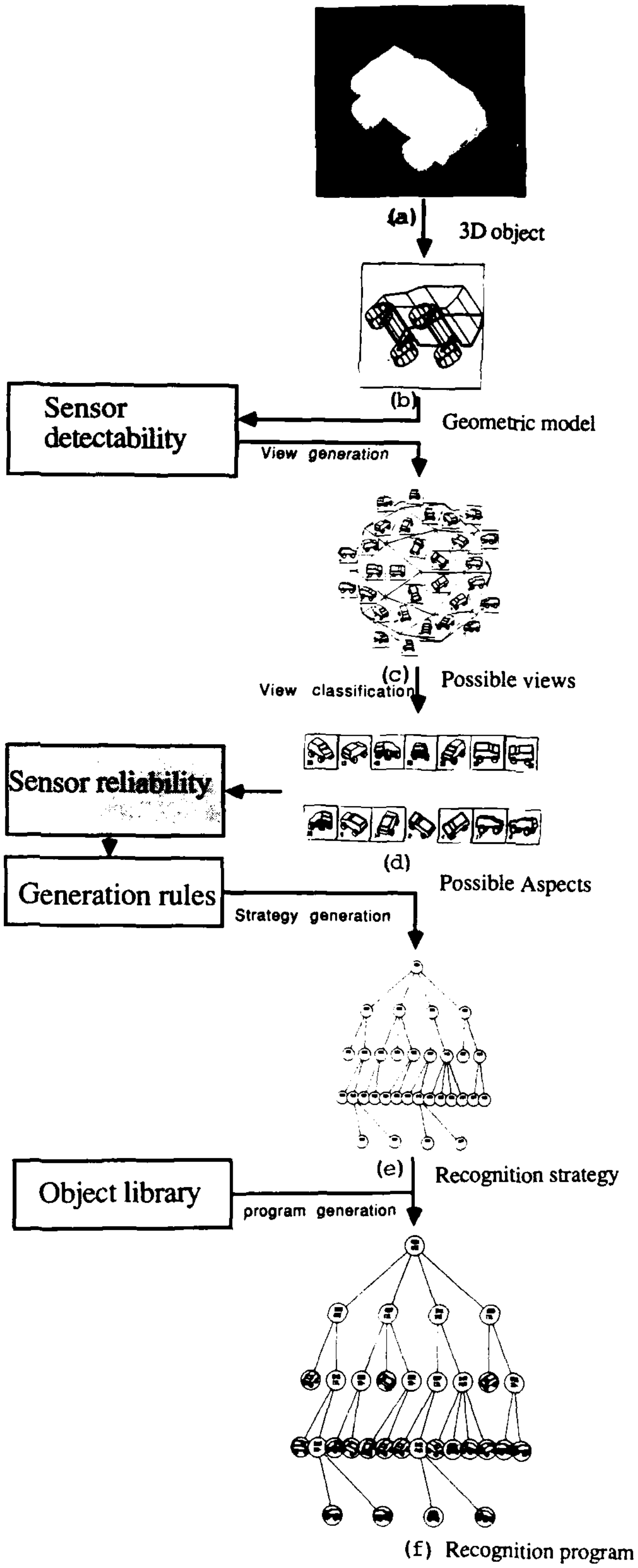


Figure 11: Sensor model and automatic generation of object recognition program.

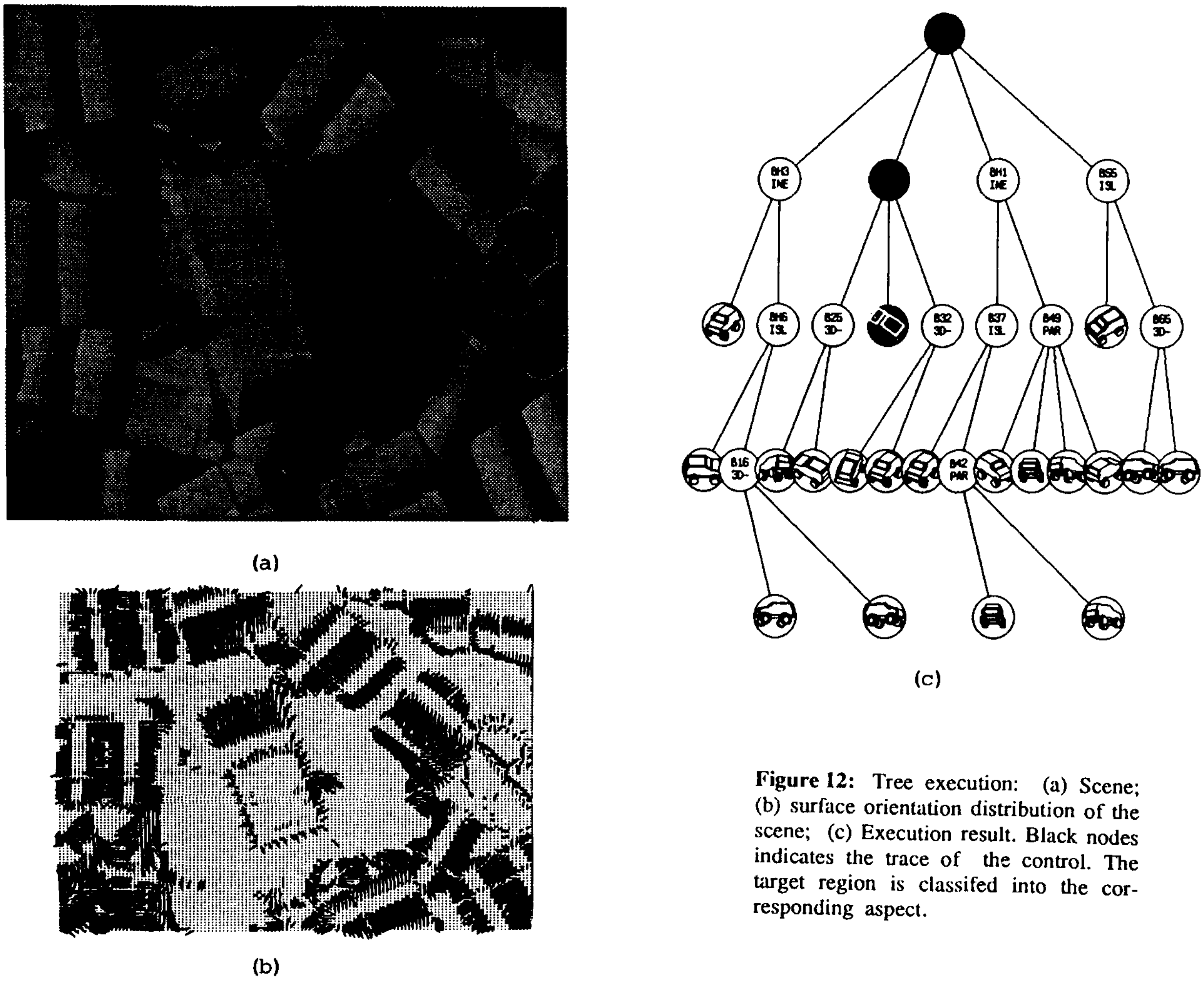


Figure 12: Tree execution: (a) Scene; (b) surface orientation distribution of the scene; (c) Execution result. Black nodes indicates the trace of the control. The target region is classifed into the corresponding aspect.

we establish the method to model the uncertainty propagation, we can also assess the uncertainty in the geometric features, hence we can construct a recognition system more systematically and reliably. Further study is required in this area.

To calculate detectable features of an object under the constraints of various sensors is a tedious job when we use a conventional geometric modeler. A better way is to interface a geometric modeler with the sensor model proposed. We call this a sensor modeler. The traditional geometric modeler only allows users to generate a 3D object by combining primitive objects and to display its views. In this sense, the traditional modeling system has only one sensor model, which is projection. The sensor modeler we propose can generate various 2D representations under given sensor specifications. Part of this facility is being implemented in our new geometric/sensor modeler VANTAGE[29].

ACKNOWLEDGEMENT

The authors thank Richard Mickelsen, Steven Shafer, Keith Gremban, Yoshinori Kuno, Huey Chang, and the member of VASC (Vision and Autonomous System Center) of Carnegie Mellon University for their valuable comments and discussions.

I. Feature coordinate system

Face

We define the z axis of the feature coordinate system to agree with the surface normal, and the x-y axes lie on the face, but are defined arbitrarily otherwise.

Edge

We define the z axis to agree with the average direction of the two normals of the two adjacent faces incident to the edge. We define the x axis of the feature coordinate system to agree with the edge direction. The y axis is determined accordingly.

Vertex

We define the z axis to agree with the average direction of the normals of the adjacent faces incident to the vertex. The x-y axes lie on the plane perpendicular to the z axis, but are defined arbitrarily otherwise.

II. G-source definition

We will define two kinds of G-sources in terms of the distribution of illuminated configurations: the uniform G-source and the directional G-source. A uniform G-source distributes its light evenly in all directions. An example of a uniform G-source is a usual light source whose illuminated configurations are located as a hemispherical cone of the source direction.

Another kind of G-source is a directional G-source which projects light depending on the rotation around the light source direction. An example of a directional source is a directional edge detector. Note that a detector is also considered as a G-source. Since the directional edge detector only detects edges with certain orientations, it is regarded as a directional source. The illuminated configurations of a directional source become a thin slice in the configuration space.

Uniform G-source

We specify a uniform G-source as

(*NS type direction angle*)

The first argument, *type*, specifies what kind of feature the G-source illuminates, and takes one of the values; *face*, *edge*, and *vertex*. The second argument, *direction*, denotes the G-source illumination direction as a vector. The third argument, *angle* defines the illuminated configurations by specifying the apex angle between the illumination direction and the z axis of the feature coordinate. If *type* is *face*, this angle defines the maximum allowable angle between the face surface normal and the illumination direction. If *type* is *edge*, this angle defines the maximum allowable angle of the smaller one of the two angles between the illumination direction and the two z axes of incident surfaces to the edge. That is, if either or both faces are well illuminated, then the edge is considered to be illuminated. If *type* is *vertex*, we have to consider at least three faces incident to the vertex. This angle defines the maximum allowable angle of the smallest angle of those angles between the illumination direction and the z axes of incident surfaces. That is, if any of the incident faces of the vertex is illuminated, the vertex is considered to be illuminated.

Directional G-source

We specify a directional G-source as

(*DS type direction angle spec–direction spec–angle*)

The first argument, *type*, specifies one of the object features: *face*, *edge*, and *vertex*. The second argument, *direction*, denotes the G-source illumination direction as a vector. The third argument, *angle*, defines the spherical angle of the illuminated configurations, as for the uniform G-source. The fourth argument, *spec–direction* defines the constraint direction to be used in the following argument. If the *type* is face, the z axis direction is used. If the *type* is edge, the x axis direction is used. If the *type* is vertex, the z axis direction is used. The fifth argument, *spec–angle* defines the maximum allowable angle between the constraint direction and the principal direction i.e. the surface normal of a face, the edge direction of an edge, and the average surface orientation around a vertex.

References

1. Binford, T.O., "Visual perception by computer", *Proc. IEEE Systems Science and Cybernetics Conf.*, IEEE, 1971.

2. Shafer, S. A. and Kanade, T., "The Theory of Straight Homogeneous Generalized Cylinders, and A Taxonomy of Generalized Cylinders", Tech. report CMU-CS-83-105, Carnegie-Mellon University, Computer Science Department, January 1983.

3. Ikeuchi, K., "Recognition of 3-D objects using the extended Gaussian image", *Proc. of 7th Intern. Joint Conf. on Artificial Intelligence*, 1981.

4. Horn, B.K.P., "Extended Gaussian Images", *Proc of the IEEE*, Vol. 72, No. 12, December 1984.

5. Pentland, A. P., "Perceptual Organization and the Representation of Natural Form", *Artificial Intelligence*, Vol. 28, No. 2, 1986.

6. Koenderink, J. J. and Van Doorn, A. J., "Geometry of binocular vision and a model for stereopsis", *Biological Cybernetics*, Vol. 21, No. 1, 1976.

7. Chakravarty, I. and Freeman, H., "Characteristic views as a basis for three-dimensional object recognition", in *Proc. The Society for Photo-Optical Instrumentation Engineers Conference on Robot Vision*, SPIE, Bellingham, Wash., Vol. 336, 1982.

8. Ikeuchi, K., "Generating an Interpretation Tree from a CAD Model for 3-D Object Recognition in Bin-Picking Tasks", *International Journal of Computer Vision*, Vol. 1, No. 2, 1987.

9. Barrow, H.G. and Popplestone, R.J., *Relational description in picture processing*, Edinburgh University Press, Edinburgh, Scotland, 1970.

10. Brady, J.M. and Asada, H., "Smoothed local symmetries and their implementation", *The International Journal of Robotics Research*, Vol. 3, No. 3, 1986.

11. Roberts, L.G., "Machine perception of three-dimensional solids", in *Optical and Electro-Optical Information Processing*, Tipplett, J.T., ed., MIT Press, Cambridge, MA, 1965.

12. Marr, D. and Hildreth, E., "Theory of edge detection", *Proc. of the Royal Society of London B*, Vol. 207, 1980.

13. Canny, J.F., "Finding edges and lines in images", Tech. report AI-TR-720, Massachusetts Institute of Technology, Artificial Intelligence Laboratory, 1983.

14. Horn, B.K.P., "Obaining Shape from Shading", in *The Psychology of Computer Vision,* Winston, P.H., ed., McGraw-Hill, New York, 1975.

15. Ikeuchi, K. and Horn, B.K.P., "Numerical shape from shading and occluding boundaries", *Artificial Intelligence*, Vol. 17, No. 1-3, 1981.

16. Tomiyasu, K., "Tutorial review of Synthetic-Aperture Rader(SAR) with applications to imaging of the ocean surface", *Proc. of the IEEE*, Vol. 66, No. 5, May 1978.

17. Jarvis, R.A., "A laser time-of-flight range scanner for robotic vision", *IEEE Tran, Pattern Analysis and Machine Intelligence*, Vol. PAMI5, No. 5, 1983.

18. Hebert, M. and Kanade, T., "Outdoor scene analysis using range data", *Proc. of Intern. Conf. on Robotics and Automation*, IEEE Computer Society, San Francisco, April 1986, pp. 1426-1432.

19. Marr, D. and Poggio. T., "A computational theory of human stereo vision", *Proc. of the Royal Society of London B*, Vol. 204, 1979.

20. Ohta, Y. and Kanade, T., "Stereo by intra- and inter-scanline search using dynamic programming", *IEEE Trans Pattern Analysis and Machine Intelligence*, Vol. PAMI-7, No. 2, 1985.

21. Milenkovic, V.J. and Kanade, T., "Trinocular vision: using photometric and edge orientation constraints", *Proc. Image Understanding Workshop*, DARPA, Miami Beach, FL, December 1985.

22. Woodham, R.J., "Reflectance Map Techniques for Analyzing Surface Defects in Metal Castings", Tech. report AI-TR-457, Massachusetts Institute of Technology, Artificial Intelligence Laboratory, 1978.

23. Ikeuchi, K., Nishihara, H.K., Horn, B.K.P., Sobalvarro, P., and Nagata, S., "Determining grasp points using photometric stereo and the PRISM binocular stereo system", *The International Journal of Robotics Research*, Vol. 5, No. 1, 1986.

24. Koshikawa, K., "A polarimetric approach to shape understanding of glossy objects", *Proc. of 6th Intern. Joint Conf. on Artificial Intelligence*, 1979.

25. R. Bajcsy, E. Krotkov, M. Mintz, "Models of errors and mistakes in machine perception", *Proc. of Image Understanding Workshop*, DARPA, 1987.

26. Faugeras, O.D., Ayache, N., Faverjon, B. and Lustman, F., "Building visual maps by combining noisy stereo measurement", *Proc. of Intern. Conf. on Robotics and Automation*, IEEE computer society, San Fransisco, April 1986, pp. 1433-1438.

27. Matthies, L. and Shafer, S.A., "Error modelling in stereo naviagtion", Tech. report CMU-CS-86-140, Carnegie-Mellon University, Computer Science Department, 1986.

28. Chang, H., "A vision algorithm generator by object-oriented programming", Tech. report, Dept. of Electrical and Computer Engineering, Carnegie Mellon University, July 1987.

29. Kanade, T., Balakumar, P., Robert, J.C., Hoffman, R., and Ikeuchi, K., "Overview of geometric/sensor modeler VANTAGE", *Proc. The International Symposuim of Exposition on Robots*, The Australian Robot Association, Sydney, Australia, November 1988.

30. Agin, G.J. and Binford, T.O., "Computer description of curved objects", *Proc. of 3rd Intern. Joint Conf. on Artificial Intelligence*, Stanford, CA, August 1973, pp. 629-640.

Rapid Object Recognition From a Large Model Base Using Prediction Hierarchies *

J. Brian Burns
Computer and Information Science Department
University of Massachusetts
Amherst, MA 01003

Leslie J. Kitchen
Department of Computer Science
University of Western Australia
Nedlands, W.A. 6009, AUSTRALIA

Abstract

An object recognition system is presented to handle the computational complexity posed by a large model base, an unconstrained viewpoint, and the structural complexity and detail inherent in the projection of an object. The design is based on two ideas. The first is to compute descriptions of what the objects should look like in the image, called *predictions*, before the recognition task begins. This reduces actual recognition to a 2D matching process, improving the efficiency of 3D object recogntion during run-time.

The second design feature is to represent all the predictions by a single, combined IS-A and PART-OF hierarchy called a *prediction hierarchy* that speeds up the indexing, or selection, of the correct object and viewpoint. The nodes in this hierarchy are partial descriptions that are common to views and hence constitute shared processing subgoals during matching. The recognition time and storage demands of large model bases and complex models are substantially reduced by subgoal sharing: projections with similarities explicitly share the recognition and representation of their common aspects.

A prototype system for the automatic compilation of a prediction hierarchy from a 3D model base is demonstrated using a set of polyhedral objects and projections from an unconstrained range of viewpoints. In addition, a recognition system is proposed that can efficiently search for matches to the object-specific goal nodes of the hierarchy.

1. Introduction

Object recognition is a central problem in computer vision. It is also a very difficult problem, both because of the vast number of potential objects that a fully general vision system must be able to recognize, and because of the multitude of appearances that each object can present when viewed from different vantage points. In addition, most practical applications would demand that object recognition be accomplished very rapidly. That humans can, at a glance, recognize an object from the tens of thousands of familiar objects [1] stands as a challenge to computer vision.

To date, significant progress has been made in model-based object recognition. (See [3,4,11,12,13] for some of the most recent efforts, and [6] for a more comprehensive review.) However, none of this work has fully addressed the problem of rapid recognition out of large model bases. Indeed, the methods used by most model-based object-recognition systems, while often reasonably successful within their own terms, could nct be scaled up to deal with large model bases, that is, those containing thousands of objects.

We present here an approach which explicitly addresses the problem of large model bases. While we do not claim to have fully solved this problem, our analysis and experience with a prototype system indicate that this approach is feasible.

2. Overview

Figure 1 shows a block diagram of the basic recognition system design. The three main modules (the boxes) are discussed in the subsections that follow.

2.1 Predictions and the Prediction Hierarchy Compiler

The prediction hierarchy compiler embodies two major aspects to our approach. The first is that before recognition, in an "off-line" process, we precompile from the 3D model base descriptions of the potential appearance of all the objects from all possible vantage points. (This is actually something of a conceptual simplification: what is actually done will be described later.) These descriptions constitute *predictions* about the 2D image appearance of the objects. By doing this we avoid, at recognition time, the cost of the complicated 3D-to-2D transformations between 3D models and 2D images. The cost of these computations in the general case is one of the chief difficulties with ACRONYM [4]. In our approach the matching at recognition time is reduced to a 2D-to-2D problem. This precomputation of predictions is possible because, while an object potentially has an infinitude of different exact appearances, only a finite number of these are different in a qualitative sense [9,14], and for many objects, the number of these qualitatively different views ("generic views") is reasonably small.

Even so, the number of such views over a large model base can still be very great. This leads to the second component of our approach, which is to organize these predictions into a hierarchical structure based on PART-OF (aggregation) and IS-A (specialization) links. This way, the common aspects of the predictions are made explicit and represented just once in the hierarchy. One advantage of this is that the set of predictions can thereby be stored more compactly. But more important, predictions common to many objects need be matched only once during the later recognition step, effecting a considerable economy in matching. Ideally, the hierarchy should act like an efficient discrimination net, permitting rapid indexing to an object-specific prediction.

*This research was supported in part by DARPA and RADC under contract F30602-87-C-0140, and by the National Science Foundation under grant DCR-8500332.

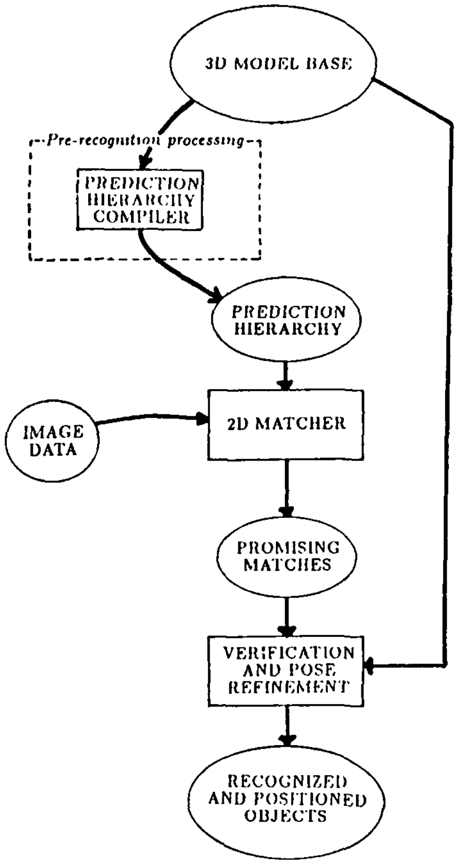

3D MODEL BASE
Pre-recognition processing
PREDICTION HIERARCHY COMPILER
PREDICTION HIERARCHY
IMAGE DATA
2D MATCHER
PROMISING MATCHES
VERIFICATION AND POSE REFINEMENT
RECOGNIZED AND POSITIONED OBJECTS

Fig. 1. The basic recognition system design.

In this way our *prediction hierarchy* provides a single, uniform mechanism for coping both with variation and similarity across objects and across views of a single object.

In reality, these two operations, of prediction and organization, are not run separately, but are intertwined. Predictions are generated and incorporated into the hierarchy incrementally, as needed. This has two related advantages. First is that a potentially unmanageable, unorganized collection of predictions is never actually created; second is that the predictions generated need only be elaborated sufficiently to ensure adequate discrimination between objects (and distinct views). The full, detailed description of the generic views of the objects normally need not be generated; only general predictions of appearance sufficient to adequately discriminate object views one from another are needed. Indeed, in the system as implemented, the concept of *generic view* does not appear at all.

Out of these twin processes of prediction and organization comes a prediction hierarchy, which encapsulates in a directly usable form the knowledge needed to visually recognize objects out of the 3D model base. Notice that, while the construction of the prediction hierarchy may be a computationally expensive process, it need be done only a single time for any given model base. Once the prediction hierarchy is compiled, it can be used over and over again for recognizing objects out of that model base.

2.2 Recognition-Time Matching

The actual recognition of an object in an image involves matching the hierarchically organized predictions against perceptual features extracted from that image. In the prototype system described here, matching takes place systematically from most-general predictions (shared by many objects) to object-view-specific predictions, thereby ensuring effective indexing from the image data into the model base.

2.3 Pose Estimation and Verification

In most applications of object recognition it is necessary not only to identify the viewed object, but also to determine its *pose* (position and orientation) with respect to the camera. Matching using the prediction hierarchy may only determine the approximate view of an object. However, it does facilitate exact determination of object pose. The correspondence between the matched prediction and the image features can be related back through the prediction to the original 3D object model, giving a correspondence between the image features and the object model. From this information can be determined the parameters of the 3D-to-2D transformation that best matches the 3D object features to their corresponding 2D image features. What is more, the approximate view associated with the matched prediction can be used as a good starting point for iterative methods of solution for the pose parameters, such as is used in Lowe's SCERPO system [13].

The exact determination of pose also provides verification of the object identification achieved by matching through the prediction hierarchy. This is important because it means that the discriminations made by the prediction hierarchy need not be perfect. It is sufficient that matching through the prediction hierarchy be conservative, that it not rule out any true object identifications. Any objects and views erroneously indexed by the prediction matching can be ruled out by the pose estimation and verification. Of course, the more false indexing ruled out by the prediction matching, the less work there is to be done by the pose estimation and verification, but there is clearly a trade-off involved here. At a certain point it may be more efficient to take a matched general prediction and verify the objects it implies one by one through pose verification, than to further extend the match to an object-specific prediction. One advantage of our approach is that this cut-off point could be determined during the prediction compilation. It is our belief that normally this cut-off point is quite late, that is, the goal nodes of the prediction hierarchy would be specific to an object-view, or at most to a few object-views.

Pose estimation for verification is related to Lowe's use of the *viewpoint consistency constraint*[13], although he uses it only in the context of single-object recognition.

2.4 The Domain

The domain we have chosen is polyhedral objects with linear surface markings. Polyhedra form a useful and rich class of objects for recognition experiments, while their 3D modeling and rendering are fairly well understood. In particular, the edges and linear surface markings of polyhedra project into straight-line segments in an image, and good techniques for extracting such straight-line features from images are readily available [2,5]. Note, however, that our overall approach is in no way restricted

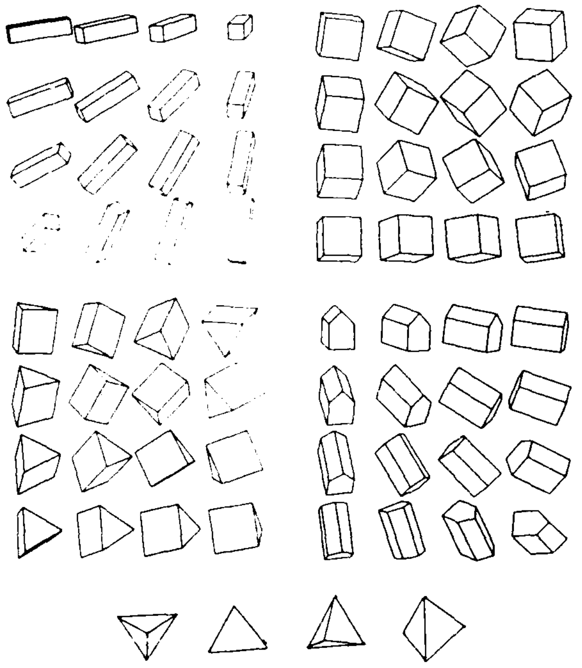


Fig. 2. 3D Objects used to demonstrate the prediction hierarchy compiler and matcher (*tall box, cube, triangular prism, house, tetrahedron*).

to polyhedra: in concurrent work at UMass, generating predictions for curved objects is also being investigated [9,10,14].

The actual objects used to demonstrate the design are shown in Figure 2. The objects have differences and similarities in various dimensions and part of the problem is to take advantage of both. The similarities in their visual structure, such as occurrences of parallelograms or of certain types of line junctions, must be utilized by the recognizer to make the search for a match efficient. The differences in visual structure, such as height-to-width proportions, must be utilized to discriminate between the objects. Additionally, the variations in visual appearance caused by variations in the camera must be taken into account while doing the structural analysis.

3. Predictions

For the construction of predictions our camera model is so-called *normal perspective*, following Brooks [4]. This is equivalent to orthographic projection with an additional image scaling factor. It provides a simple but useful approximation to true perspective projection so long as the camera is not too close to the object. (Note that the use of normal perspective for predictions does not preclude using true perspective for the pose estimation and verification step.)

A *prediction* is a statement concerning some structural aspect of the image of an object. For example, this may be as simple and general as an assertion that there exists a pair of parallel segments in the projection; or as complex as a description of an image unique to some object. A prediction is represented here as a relational graph; the elements in the graph are projected straight-line segments. The relations associated with arcs in the graph mutually constrain the relative orientations, positions and sizes of pairs of line segments. For example, the relation *parallel* constrains two segments to having the same orientation. The predictions can be viewed as conjunctions of these relations over formal segment arguments.

The objects are expected in any pose, hence all predictions generated are invariant to re-scaling, translation and rotation in the image plane. Because of this and the type of projection assumed, there are only two degrees of camera variation that have a significant effect on the projections being described by the predictions: the two angular components of the viewpoint that sweep out a *viewing sphere* about the object.

Any given prediction may be satisfied by the projection of more than one object in the model base and, in fact, it may be satisfied by the projection of more than one set of line segments on the same object. This is especially true for simple structural statements, such as the parallelogram prediction in Figure 3. For each such occurrence, or *instance*, of a prediction, there may be a wide range of viewpoints for which the set of projected line segments satisfy the prediction. For example, the projections of the 3D line segments a, b, c, and d of the triangular prism model (Figure 3), satisfy the parallelogram prediction over all the viewpoints for which these segments are simultaneously visible (half the viewing sphere).

More specifically, we define a prediction *instance* as a one-to-one mapping from a set of projected 3D model segments to the elements of the prediction's relational graph and the set of

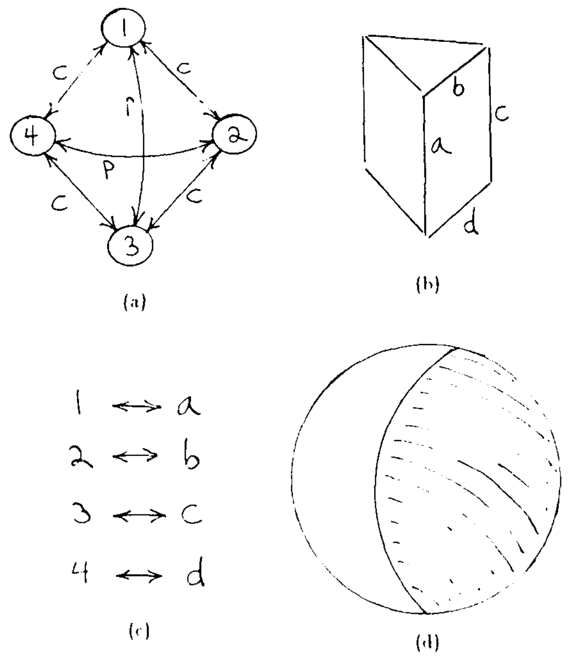
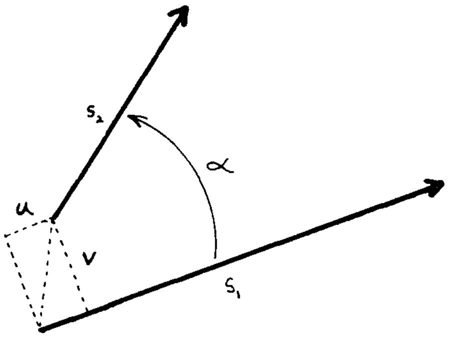


Fig. 3. Example of a prediction *instance*: (a) the prediction's relational graph (c arcs = "coincident endpoints" and p arcs = "parallel"); (b) 3D model segments; (c) mapping between prediction elements and projected model segments; and (d) range of views for which mapping satisfies prediction (shaded region).



$$min\_u \le u \le max\_u$$
$$min\_v \le v \le max\_v$$
$$min\_\alpha \le \alpha \le max\_\alpha$$
$$min\_s \le s \le max\_s$$

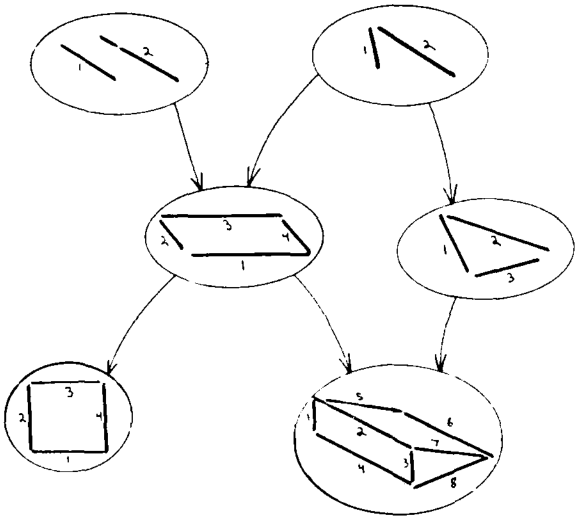
Fig. 4. Representation of image segment relations.

viewpoints (the *view region*) on the viewing sphere from which the prediction is valid for these segment-element bindings. For a given model base, there is a set of such instances associated with each prediction. During the compilation of the prediction hierarchy, it is important to keep track of all instances of a prediction and their associated view regions.

A relation between a pair of projected segments used in the predictions is represented by ranges of four relational measures, u, v, α and s (see Figure 4). Let us consider two segments, s_1 and s_2. The vector (u, v) is the position of segment s_2 relative to s_1: it is the displacement of an endpoint of s_2 from an endpoint of s_1 measured along s_1 and normal to s_1, divided by the length of s_1. The angle between them, α, is measured counterclockwise from s_1 to s_2; and s is the relative scale or length ratio of s_2 over s_1. A relation is defined as an *extent box*, i.e. intervals in u, v, α and s, in order to capture the variation over ranges in viewpoint. For instance, projecting a pair of parallel object segments over all possible viewpoints will generate a set of measurements that have a single value (zero) in the α dimension, but some non-trivial extent in the others. Similarly, the family of projections of a pair of object segments that share an endpoint can be represented by a relation that has the value zero in both position components (u, v).

A relation between projected segments is considered useful if it is valid over a wide range in viewpoints and its extent box is small in volume (for example, consider the two view-invariant relations just mentioned). The latter property is important if the relation is to help in the indexing process; that is, to characterize an object's projection with a specificity sufficient to discriminate the object from a large number of other objects and from chance arrangements of image segments. Although invariant relations are clearly useful [13], alone they are not in general sufficient to fully characterize projections. For instance, proportions are often strong characterizations of object structure, but the length measurement ratios that represent them are often not strictly view invariant. For example, the tall box in Figure 2 has a height to width ratio that is significantly different from the cube over a large range in views and no other property can be used to differentiate them.

Any given prediction in the *prediction hierarchy* has an associated relational graph, but explicitly it is almost always represented as some combination or specialization of other predictions (see Figure 5). A prediction is a *specialization* of another if it can be described by adding new relations or narrowing the extent boxes of existing ones (i.e., tightening the constraints). For example, the rhombus can be described by adding a relation between the bottom and side segments of the more general parallelogram prediction that constrains the ratio of their length to be one. A prediction is a *combination* of other predictions if it can be described as a conjunction of these other predictions. Predictions may be combined in various ways, depending on the mappings between the relational graph elements of the part predictions and those of the whole combination. Figure 6 shows an example of how the nature of the combination prediction can



```
(define parallelogram (s1 s2 s3 s4)
  (and (coincident s1-head s2-head)
       (coincident s2-tail s3-head)
       (coincident s1-tail s4-head)
       (coincident s3-tail s4-tail)
       (parallel s1-tail s3-tail)
       (parallel s2-tail s4-tail)
       ))
(define triangle (s1 s2 s3)
  (and (coincident s1-head s2-tail)
       (coincident s2-head s3-tail)
       (coincident s3-head s1-tail)
       ))
(define triangular-prism (s1 s2 .. s8)
  (and (parallelogram s1 s2 s3 s4)
       (parallelogram s5 s2 s6 s7)
       (triangle s3 s7 s8)
       ))
(define rhombus (s1 s2 s3 s4)
  (and (parallelogram s1 s2 s3 s4)
       (same-size 1 2)
       ))
```

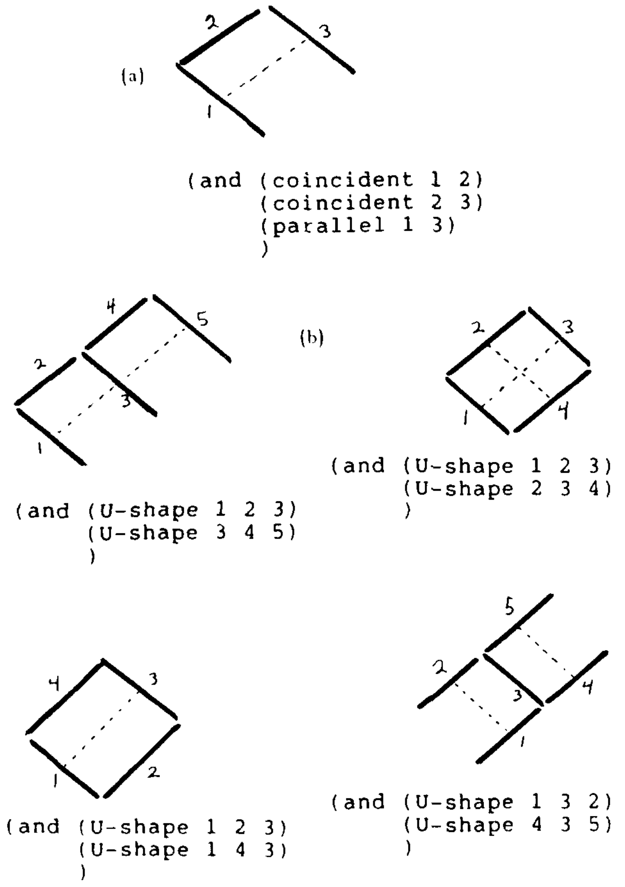
Fig. 5. Predictions as specializations or combinations of other predictions. (Multiple links between nodes not shown.)

change by varying these *part-whole* maps, which are represented by the arguments passed into the part predictions and the ordering of these arguments.

Note that, because of the combination predictions, the "hierarchy" is actually a directed acyclic graph, where the predictions are *nodes*, and the specializations and combinations are *successors* of the nodes they are created from (their *predecessors*). For the prototype system presented in this paper, the hierarchy compiler and matcher have only been developed to use the combination process. A compiler that uses specialization will be presented in forthcoming work [8]. Also, the compiler currently generates combinations that are limited to pairs of parts.

4. Prediction Hierarchy Compilation

In Section 3 it was shown that a prediction hierarchy is composed of predictions that are combinations or specializations of other predictions. The primitive, or base-level predictions of the hierarchy (those that all the other predictions are derived from) ought to be those that are valid for very large numbers of objects and are essentially invariant to viewpoint; such predictions are like the invariant binary relations discussed above and in [13]. The final or *goal* predictions ought to be valid for only a few objects and over a narrow enough range of viewpoints that pose estimate refinement and verification can be performed effectively.



```
(and (coincident 1 2)
     (coincident 2 3)
     (parallel 1 3)
     )

(and (U-shape 1 2 3)
     (U-shape 3 4 5)
     )

(and (U-shape 1 2 3)
     (U-shape 2 3 4)
     )

(and (U-shape 1 2 3)
     (U-shape 1 4 3)
     )

(and (U-shape 1 3 2)
     (U-shape 4 3 5)
     )
```

Fig. 6. An example of how different part-whole maps can create different combinations. (a) *U-shape* prediction. (b) predictions created by the combination of *U-shape* with itself under different part-whole maps. These are represented by arguments passed into the predecessor prediction (*U-shape*) and the ordering of these arguments.

A useful way to build such a structure is to start with a small set of simple, ubiquitous predictions and then iteratively search for useful combinations or specializations, eventually isolating predictions that characterize the projections of specific objects. This builds the hierarchy in the direction in which the indexing process proceeds through it, generating successors that seem most useful for discriminating the possibilities associated with each current node.

Though the major concern of this work is the efficiency of the recognition process, the pre-recognition compiler design should also avoid combinatorial explosions that, for a fair-sized model base, would keep it running until the next century. By using this iterative construction approach, we limit the combinatorial complexity, and hence processing time, to a manageable level. This is because the system is never performing isomorphism analysis over large graphs: it is always comparing combinations of small numbers of parts.

4.1 Criteria for Selecting Successors

In building the prediction hierarchy, it is important to understand which of the possible combinations and specializations of simpler predictions are useful to explicitly represent as nodes in the hierarchy. For maximum efficiency in the search for the exact object and view, the hierarchy should be in the form of a binary search structure. To achieve this, each new node added by the compiler should be satisfied by exactly half of the instances of its predecessors (where instances are the possible segment-element maps of a prediction described in Section 3). This would make the new node an ideal test in a binary search process.

This is roughly what the compiler strives to build, though there is an additional consideration that changes the picture somewhat. The *absence* of a match to a prediction is not considered here as evidence for the presence of alternative objects or views. For each alternative subset of possibilities (instances), there should be an associated successor that requires additional positive evidence in the image in order to be satisfied, i.e., some new image part that needs to be present. This is because the absence of a match could be due to noise, instead of the absence of the object; and noise is more likely to prevent matches than to create spurious matches to a non-trivial relational graph description. For example, if the recognizer has reached node 8 in Figure 7 and is trying to decide whether the object projected is a triangular prism or something else, the absence of the triangle match does not necessarily mean that it is a projection of some other object, but the presence of an additional parallelogram in place of the triangle is strong evidence that it is indeed another object.

Thus, the compiler does not try to add predictions whose satisfaction acts as a discrimination test. Instead, the compiler associates each subset of instances with a distinct combination successor. The number of successors per prediction (i.e., the branching factor of the hierarchy) should be kept low since large branching factors can slow the match search process down. In addition, it is important that every instance of a prediction be associated with *some* successor of this prediction. Whenever there is a match to a non-goal prediction, there should also be a match to at least one of its successors, so that there can be some path, via a sequence of matches, that culminates in a goal prediction match. An instance is *covered* if there is a successor that is also satisfied by the same same object, same view and compatible segment-element bindings (given the part-whole maps relating predecessor elements to successor elements).

In the ideal case then, there would be two successors per node, each with positive evidence (such as the triangle or parallelogram above), and togethur they partition the prediction instances exactly in half. This ideal case cannot always be met: there may not be successors that are satisfied by half of the instances, but only by somewhat smaller or larger fractions; and there may not be a set of successors that partition the instances into disjoint subsets. Nevertheless, at each iteration the compiler should try to select as small a set of successors as possible that still cover all the instances of the predictions generated the iteration before.

4.2 Iterative Hierarchy Construction

For the experiment reported here, *parallelism* and *endpoint coincidence* are used as the initial set of primitive predictions since they occur frequently in the objects studied here, they can be rapidly matched and their combinations can be used to discriminate between most of the objects and views. The iterative construction process stops when all of th redictions without successors are either associated with single objects or, if they are satisfied by projections of more than one object, there are no differences between their projections that the compiler is capable of representing.

For each iteration of the combination process, the system isolates useful combinations by (1) finding predictions that often appear together in the same projections, and then (2) selecting a subset of these commonly occurring combinations that approximately satisfy the criteria discussed in Section 4.1

The co-occurrences between predictions are found by noting the amount of overlap in the view regions of their instances. All instances of all predictions are stored in data structures called *visibility maps*. There is a visibility map for each object; the maps are arrays of cells indexed by the two viewpoint parameters, making a discrete sampling of the view sphere about the object. Each cell lists prediction instances visible from the associated viewpoint and object; and with each prediction is a list of cells that contain it. To find frequent co-occurences between some prediction p and other predictions, the system looks for predictions that appear in the same cells as p and accumulates the total number of cells for each that do.

Besides visual co-occurrence, an additional constraint is added to the pairs of predictions considered for combination. In order for the resulting successor prediction to a be a spatially compact and connected structure (instead of corresponding to widely separated parts of the projection of the object), only pairs that share at least one projected line segment are considered. For example, the two parallelogram parts of the triangular prism projection in figure 5 share a segment.

After finding all commonly occurring combinations at the current level, the compiler tries to select a small subset that collectivel nd efficiently cover the instances of the predictions that they are successors of. The successors are iteratively selected in the following way:

1. Select the successor that covers the largest number of previously uncovered instances.

2. Update the records of all the instances just covered by this new successor.

Note that this iterative selection process is an inner loop to the iterative process that is building up the prediction hierarchy level by level. This inner loop works within a level to select a useful set of successors at that level.

4.3 Example Results

The prediction hierarchy compiler was run on the model base of polyhedral objects shown in Figure 2. Specialization was used in only a rudimentary way, to distinguish the tall box from the cube at the very end. All the other predictions were formed by combination.

The resulting hierarchy is shown diagrammatically in Figure 7. As can be seen, it is a quite reasonable representation of visual knowledge about these objects, organized for efficient recognition. Other prediction hierarchies are also possible given

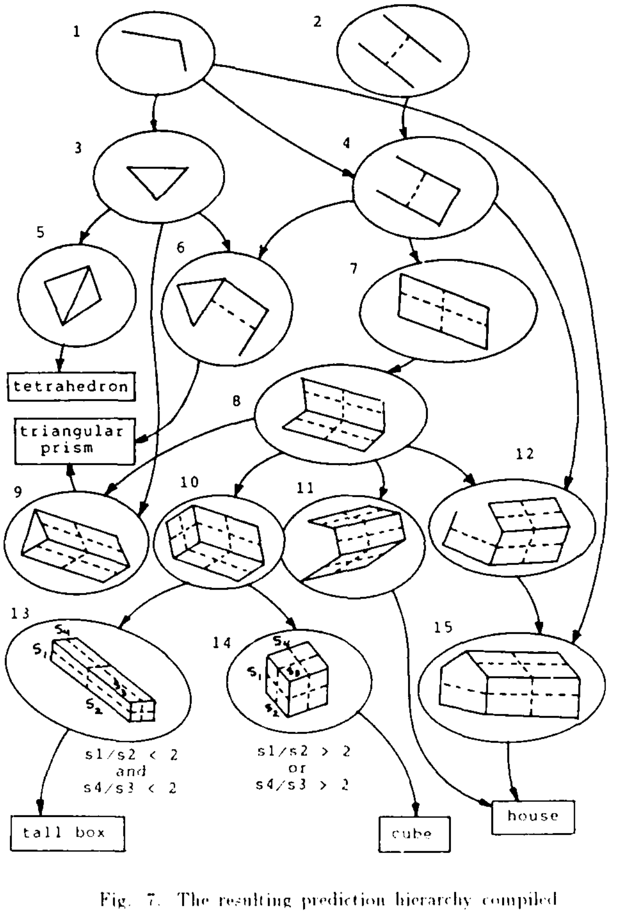


Fig. 7. The resulting prediction hierarchy compiled from views of the objects in figure 2. The nodes represent predictions and arrows indicate combination and specialization links. The predictions are represented graphically by segments and dashed lines for parallel relations

the objects modeled. For example, node 6 of Figure 7 could alternatively have been the combination of a triangle with a parallelogram, rather than the *U*-shaped structure. This might actually be preferred for greater noise insensitivity, since the parallelogram could afford to lose more line segments to noise and still distinguish the triangular prism from the other objects. Since the compiler is a single-pass algorithm, only combining predictions that have already have been generated, it noticed the triangle and *U*-shape combination before the parallelogram was constructed. It may be possible to add a post-processing step that realizes more intelligent combinations than can be isolated in a single-pass system.

5. Matching

This section is a brief summary of the matching system currently under development. The system tries to find matches to the object-specific goal predictions given a prediction hierarchy and matches to the primitive, base-level predictions. Starting with these readily found base-level matches, the basic idea is to repeatedly attempt to extend existing matches into matches of more complex successor nodes until good matches to goal nodes are achieved.

The following problems are being investigated in the process of designing this system:

1. Missing straight-line segments in the image, or those "not seen" by the matcher because they are mis-measured, can block match extensions that are steps towards a goal node match.

2. For a given match, there may be a large tree of direct and indirect successors of the node matched. The matcher should avoid exhaustively searching this tree for matches to successors.

3. Because predictions are often represented as combinations of predecessors that may themselves be large and complex, the attempt to extend a match to a successor match can involve a costly search for the other predecessor. These searches should be avoided, or methods of reducing their cost should be implemented.

4. Image clutter can generate a lot of possible match extensions, potentially slowing down the search for goal matches.

5. Symmetries in a node's relational graph description can slow the search for matches to its successors. There may be many possible locations to check for the match of the other predecessor. There also can be ambiguities to resolve in the image-successor match bindings, given the matches to the predecessors.

We believe that a system of the following basic design can handle these problems. It is a set of asynchronous matching processes that are distributed spatially about the image. Each process repeatedly attempts to jointly extend *pairs* of existing matches whose image segments lie in a spatial zone associated with the process. The extension operation has the following steps:

- Select a pair of existing matches whose image segments have some non-accidental relationship, such as those discussed in [13], or matches that actually share an image segment. Segment sharing can be considered a certain kind of non-accidental relationship: the identity relation.

- Search the prediction hierarchy for common - and possibly indirect - successors to the pair of nodes matched. This is done using a marker passing and collision detection scheme.

- Attempt to complete a match to the common successor given the pair of predecessor matches and the part-whole maps (see Section 3) between their relational graph elements. The completion process may involve a search for particular image segments if the common successor is indirect.

For typical images, a number of the expected straight-line segments are missing or mis-measured. In spite of this incomplete information, the objects are often readily recognizable. For a recognition system to detect objects in such data, it must have some ability to find and accept good partial matches, where the quality of the partial match is some function of the fraction of well-matched segments. Because the absence of segments may prevent a good partial match of a direct successor, but not to an *indirect* successor, the system should be capable of extending a node's match to one of its indirect successors. For example,

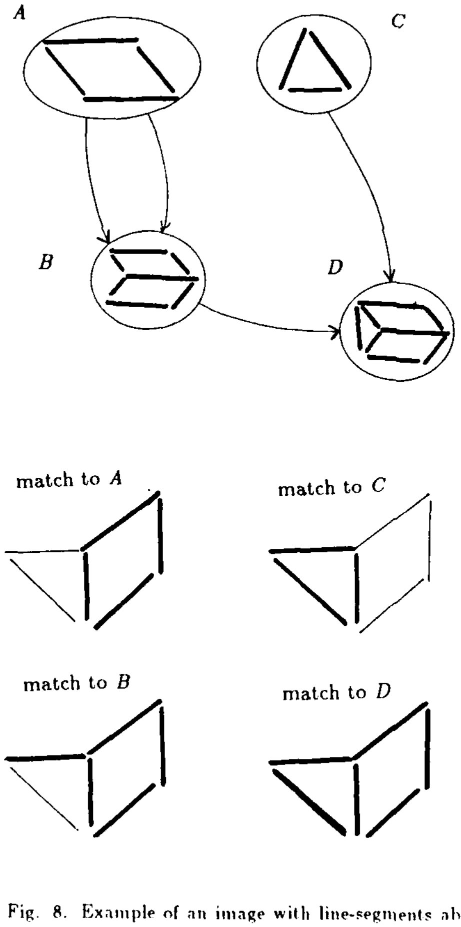


Fig. 8. Example of an image with line-segments absent in such a way that an indirect successor (*D*) of a node (*A*) has a better match than the direct successor between them (*B*) given that match quality is in terms of the fraction of the prediction elements matched. Matched image segments are shown in bold.

extending the match of A in Figure 8 to a satisfactory match to B may be impossible; but a match to D, an indirect successor of A, may be good enough in spite of the missing segments.

Each node may have a potentially large tree of successors, and matching to a successor can be involved. Thus the attempt to find matches to direct or indirect successors should not be an exhaustive search of the tree of possible successors. This is where the strategy of attempting to jointly extend a *pair* of existing matches helps: it focuses the search to intersections of the successor trees of the matched nodes. These intersections tend to involve much smaller numbers of successors, and their matches are more promising since *two* predecessors have already been matched. Joint extension of pairs of matches is especially useful when the nodes have symmetries. In these cases, a thorough search for a successor, given a match to only one predecessor, may involve checking many image locations for the match of the other, missing predecessor.

Pursuing joint extensions of pairs of matches does help reduce the amount of effort the matcher spends in the costly search for complex predecessor matches; but, for the system to rely on joint extension, pairs of jointly-extendible matches have to be generated. Also, both members of such a pair should be generated within a reasonable time frame (i.e., there should not be a long delay between their creations). By following a simple depth-first policy, the matcher may often attempt a large number of alternative matches to roughly the same image data, and thus delay the generation of matches over other image data. This is undesirable because predecessors of a node tend to represent parts of the node's relational graph description that are spatially distributed. In other words, jointly-extendible matches may share some image segments, but they tend to occupy different portions of the image. Thus, if matches are being attempted in different parts of the image at roughly the same time and level in the hierarchy, there is a better chance for jointly extendible pairs of matches to occur and hence a more rapid convergence to a goal match. This can be done by explicitly allocating the system resources to matching in distinct spatial zones distributed about the image. This should be done whether or not the hardware is capable of doing these matching operations in parallel.

A fair number of matches to intermediate nodes may be generated in a complex, cluttered image during the pursuit of matches to goal nodes, and thus the number of possible *pairs* of matches to which joint extension can be attempted could be large. The answer is to restrict the joint extensions to pairs that are much more likely to be part of the same object's projection. In complex images, this could dramatically reduce the number of pairs attempted, and these pairs have a much greater chance of being extended. A good heuristic for doing this is to select pairs of matches whose image segments have what Lowe[13] describes as associations that are not likely occur by accident. For example, the matches of A and C in Figure 8 share an image segment, which almost always occurs when the matches are to projections of the same object.

6. Conclusions

We have described an approach to rapid object recognition from large model bases. A structure called a *prediction hierarchy* is used to organize knowledge about the visual appearance of objects in a form directly usable for rapid indexing of the model and general view. We present a prototype implementation of the prediction hierarchy compiler and show its results using a small model base of polyhedral objects. In addition, a recognition system is proposed and discussed that can efficiently search for matches to the object-specific goal nodes of the hierarchy.

Current work [8] includes further development of the recognition system and a prediction hierarchy compiler capable of specialization. Future directions for this research include a multipass prediction hierarchy compiler that is more intelligent in its selection of successor nodes and also the compilation of sophisticated control strategies into the hierarchy.

Acknowledgments

Special thanks to Ed Riseman, Allen Hanson, Michael Boldt and John Dolan for comments, support and help with the figures.

REFERENCES

[1] Biederman, I., "Human image understanding: Recent research and a theory", *Computer Vision, Graphics, and Image Processing*, vol. 32, pp. 29-73, 1985.

[2] Boldt, M. and R. Weiss, "Geometric Grouping of Lines", CVPR, pp. 489-495, 1986.

[3] Bolles, R. C. and R. A. Cain, "Recognizing and locating partially visible objects: The Local-Feature-Focus Method", *International Journal of Robotics Research*, vol. 1, no. 3, pp. 57-82, 1982.

[4] Brooks, R. A., "Symbolic reasaoning among 3D models and 2D images", *Artificial Intelligence*, vol. 17, pp. 285-348, August 1981.

[5] Burns, J.B., Hanson, A.R. and E.M. Riseman, "Extracting Straight Lines", IEEE Trans. Pattern. Anal. Mach. Intell., v.8, no.4, pp. 425-455, Jul. 1986.

[6] Burns, J. B. and L. J. Kitchen, "An approach to recognition in 2D images of 3D objects from large model bases", Technical Report 87-85, COINS Dept, University of Massachusetts, Amherst, MA 01003, August 1987.

[7] Burns, J. B. and L. J. Kitchen, "Recognition in 2D images of 3D objects from large model bases using prediction hierarchies", Proc. IJCAI-10, 1987.

[8] Burns, J. B., forthcoming Phd Dissertation, University of Mass., Computer and Information Science.

[9] Callahan, J. and R. Weiss, "A Model for Describing Surface Shape", CVPR, pp. 240-245, 1985.

[10] Dolan, J., "The Perceptual Organization of Image Curves", Technical Report, in preparation, COINS Dept, University of Massachusetts, Amherst, MA 01003, 1988.

[11] Draper, B., Collins, R., Brolio, J., Griffith, J., Hanson, A. and E.M. Riseman, "Tools and Experiments in the Knowledge-Based Interpretation of Road Scenes", *Proceedings of the DARPA Image Understanding Workshop*, January 1987.

[12] Hanson, A.R. and E.M. Riseman, "The VISIONS Image Understanding System - 1986", in *Advances in Computer Vision*, Chris Brown, Ed., Erlbaum Press, 1987

[13] Lowe, D. G., "The viewpoint consistency constraint", *International Journal of Computer Vision*, vol. 1, pp. 57-72, 1987.

[14] Weiss, R. and J. Callahan, "Generic views of piece-wise smooth surfaces", Technical Report, in preparation, COINS Dept, University of Massachusetts, Amherst, MA 01003, 1987.

EVALUATION OF QUANTIZATION ERROR IN COMPUTER VISION

Behrooz Kamgar-Parsi
Behzad Kamgar-Parsi

Computer Vision Laboratory
Center for Automation Research
University of Maryland
College Park, MD 20742-3411

ABSTRACT

Due to the important role that digitization error plays in the field of computer vision, a careful analysis of its impact on the computational approaches used in the field is necessary. In this paper we develop the mathematical tools for the computation of the average error due to quantization. They can be used in estimating the actual error occurring in the implementation of a method. Also derived is the analytic expression for the probability density of error distribution of a function of an arbitrarily large number of independently quantized variables. The probability of the error of the function to be within a given range can thus be obtained accurately. In analyzing the applicability of an approach one must determine whether the approach is capable of withstanding the quantization error. If not, then regardless of the accuracy with which the experiments are carried out the approach will yield unacceptable results. The tools developed here can be used in the analysis of the applicability of a given algorithm, hence revealing the intrinsic limitations of the approach.

1. INTRODUCTION

When an image is taken a large scene is mapped onto a plane of small dimensions. This compression of the scene into a small plane together with the fact that the coordinates of the points in the image plane are quantized introduces a serious handicap in the computational approaches used in computer vision. Quantization of the the image plane is necessary, because otherwise the image cannot be processed by a digital computer. The limited resolution capabilities of the receiver (which may be interpreted as the limited number of sensory elements in the digital computer) requires the entire scene to be mapped on an m by n array of points. Such an array is usually much too small to adequately represent the world in front of the camera. As a result a significant amount of error is introduced into computations that involve the locations of image points and features.

Spatial quantization is not the only kind of digitization in computer vision. As the number of levels of brightness that the receiver can distinguish is limited, the brightness level at a given pixel in the image plane is also digitized into, say, k values. The digitization of the brightness level, too, is a serious drawback for reliable computations in many areas of computer vision.

Due to the important role that digitization error plays in the field of computer vision, a careful analysis of its impact on the computational approaches used in the field is called for. Such analysis can serve two purposes: a) It helps us learn how much the results of the computations are affected by this error. b) It can provide insight on how to reduce the impact of this error. Note that unlike most other types of error, digitization error cannot be reduced by performing more accurate experiments. Rather it is caused by the inherent limitations of the instruments used in image processing. Therefore, a careful analysis of digitization error can reveal the intrinsic limitations of a given approach or algorithm.

In analyzing the applicability of an approach (which, of course, is assumed to be sound theoretically), one must know how susceptible to error that approach is. To evaluate how error prone a given method is, one must have a realistic measure of the error. A measure of error, for example, may be the maximum error that can occur in applying an approach, namely a worst case analysis of the approach. In general, however, knowledge of the maximum error is of limited value, because frequently the actual error that occurs in the implementation of an approach is far less than the maximum error which could have occurred. The maximum error is a gross exaggeration of the error and should not be used in the evaluation of the applicability of a method unless the likelihood of its occurrence is significant. In evaluating the reliability of an approach in practice, a much better measure of error is the average error.

Also, it is very useful to know the distribution of the error. That is, in addition to the average error, one would like to know what is the probability for the error to be within a given range. For example, is the probability of the error being larger than a certain threshold significant, or is it negligible?

2. FORMULATION OF THE AVERAGE ERROR

The maximum quantization error in quantity A being quantized is half the size of the quantization unit, γ. (For example, in spatial quantization the maximum error is half a pixel.) That is to say, the actual value, A_a, of the quantity A can be anywhere in the interval $I = [A_q - \frac{\gamma}{2}, A_q + \frac{\gamma}{2}]$, where A_q is the value of A after quantization. Furthermore, the likelihood of the actual value of A, i.e. A_a, being at a certain place within the interval follows a uniform probability density. In other words, the probability of A_a lying inside the small interval dA around the point A_q within I is independent of A and equal to dA/γ. Thus, the average value of the digitization error in A is given by

$$\frac{1}{\gamma}\int_{A_q-\gamma/2}^{A_q+\gamma/2} dA|A-A_q| = \frac{1}{\gamma}\int_{-\gamma/2}^{\gamma/2} dx|x| = \frac{\gamma}{4}. \qquad (1)$$

If A is a function of several quantities each of which is quantized, then the situation is more complicated. Let $A = f(A_1 A_2 \ldots A_N)$, where the actual value, A_{ia}, of the quantity A_i can be anywhere in the interval $I_i = [A_{iq} - \frac{\gamma_i}{2}, A_{iq} + \frac{\gamma_i}{2}]$ with equal probability. As before A_{iq} is the value of A_i after quantization. To obtain the error in A due to quantization errors in the A_i's, we consider the Taylor series expansion of A, retaining only the first order terms in the quantization error of independent variables $\Delta A_i = A_i - A_{iq}$:

$$\Delta A \simeq \frac{\partial f}{\partial A_1}\Delta A_1 + \cdots + \frac{\partial f}{\partial A_N}\Delta A_N. \qquad (2)$$

This expansion is valid when ΔA_i or equivalently (since $|\Delta A_i| \leq \gamma_i/2$) the interval I_i is small compared to A_{iq}. When the interval I_i is not small, higher order terms in the expansion must also be taken into account. However, here we assume that the linear terms in ΔA_i are sufficient. This is the only approximation we make in this paper. Obviously if higher order derivatives of $A = f(A_1 \ldots A_N)$ are small or vanish, then expression (2) remains valid even if the interval I_i is not small.

The maximum error in A, i.e. E_{max}, is thus given by

$$E_{max} = |\frac{\partial f}{\partial A_1}E_{1,max}| + \cdots + |\frac{\partial f}{\partial A_N}E_{N,max}| = \sum_{i=1}^{N}|\alpha_i\gamma_i/2|, \quad (3)$$

where $E_{i,max} = \gamma_i/2$ is the maximum error in A_i, and the coefficient $\alpha_i = \frac{\partial f}{\partial A_i}$ is evaluated at $A_{1q} \ldots A_{Nq}$.

The most realistic measure of the quantization error is the average error $\overline{E}$, which is the average of the absolute value error $E = |\Delta A|$, given by

$$\overline{E} = \frac{1}{\gamma_1 \cdots \gamma_N}\int_{-\gamma_1/2}^{\gamma_1/2} dx_1 \ldots \int_{-\gamma_N/2}^{\gamma_N/2} dx_N |\sum_{i=1}^{N}\alpha_i x_i|, \qquad (4)$$

where similar to (1) x_i stands for $x_i = A_i - A_{iq}$.

Because the evaluation of the average error for an arbitrary error distribution is not possible, often for convenience the standard deviation, σ, is used as the measure of error[1]. The standard deviation for quantization is defined through

$$\begin{aligned}\sigma^2 &= \tfrac{1}{\gamma_1\cdots\gamma_N}\textstyle\int_{-\gamma_1/2}^{\gamma_1/2} dx_1 \ldots \int_{-\gamma_N/2}^{\gamma_N/2} dx_N (\Delta A)^2 \\ &= \tfrac{1}{12}(\alpha_1^2\gamma_1^2 + \cdots + \alpha_N^2\gamma_N^2),\end{aligned} \qquad (5)$$

where $\sqrt{\gamma_i^2/12}$ is the standard deviation of the random variable x_i with uniform distribution in the interval $[-\gamma_i/2, \gamma_i/2]$. For the very special case when $N \to \infty$ (so that the Central Limit Theorem absolutely holds) and all the products $\alpha_i\gamma_i$ are identical, the simple relationship $\sigma/\overline{E} = \sqrt{\pi/2}$ exists between the average error and the standard deviation[2], however, in general no particular relationship between the two is known. Also, as far as we know a derivation of (4) good for quantization and instrumental error has not appeared in the literature.

The average error integral (4) can be evaluated approximately using the Central Limit Theorem[3,4]. This approximation works well only when N is large and all the α_i are of the same order of magnitude. In the next section we calculate the average error integral exactly.

From now on without loss of generality we replace γ by 1. It will be understood that if the error is due to the digitization of the intensity level, the limits of the integral will be $\pm 1/2$ units of gray level; if the error is due to spatial quantization, the limits of the integral will be $\pm 1/2$ pixels.

A simple example illustrating the application of the above integral is as follows. Consider the horizontal distance A between two pixels (A_1, y) and (A_2, y) in the image plane. Thus $A = A_1 - A_2$ which gives $\alpha_1 = 1$ and $\alpha_2 = -1$. Hence the average error in the distance A is

$$\overline{E} = \int_{-1/2}^{1/2} dx_1 \int_{-1/2}^{1/2} dx_2 |x_1 - x_2| = \frac{1}{3} = 0.33, \qquad (6)$$

while the maximum error is $E_{max} = |\frac{1}{2}| + |-\frac{1}{2}| = 1$ and the standard deviation is $\sigma = 1/\sqrt{6} = 0.41$ pixel.

3. CALCULATION OF THE AVERAGE ERROR INTEGRAL

In this section we calculate the average error integral

$$\overline{E}(\alpha) = \int_{-1/2}^{1/2} dx_1 \ldots \int_{-1/2}^{1/2} dx_N |\alpha_1 x_1 + \cdots + \alpha_N x_N|, \quad (7)$$

for arbitrary N and α. Here $\alpha = (\alpha_1, \ldots, \alpha_N)$. This integral is independent of the sign of α_i. In order to show this we make the transformation $x_i \to -x_i$ and obtain $\overline{E}(\ldots, \alpha_i, \ldots) = \overline{E}(\ldots, -\alpha_i, \ldots)$. Hence, we assume that all $\alpha_i > 0$. We order the α_i such that $\alpha_1 \geq \alpha_2 \ldots \geq \alpha_N \geq 0$, with at least $\alpha_1 \neq 0$; otherwise the integral vanishes.

We would like to remove the absolute value sign in the integrand of (7); to this end we introduce the Heaviside step function

$$\Theta(t) = \begin{cases} 0 & \text{if } t \leq 0 \\ 1 & \text{if } t > 0 \end{cases} \qquad (8)$$

This step function satisfies the identity

$$\Theta(t) + \Theta(-t) = 1 \qquad \text{for all } t. \qquad (9)$$

With the help of step functions we can express an arbitrary function $f(|t|)$ as

$$f(|t|) = f(t)\Theta(t) + f(-t)\Theta(-t). \qquad (10)$$

Thus we may write (7) as

$$\overline{E}(\alpha) = \int_{-1/2}^{1/2} dx_1 \ldots \int_{-1/2}^{1/2} dx_N [A\Theta(A) - A\Theta(-A)], \qquad (11)$$

where

$$A = \alpha_1 x_1 + \cdots + \alpha_N x_N \qquad (12)$$

We make the transformation $x_1 \to -x_1, \ldots, x_N \to -x_N$ in the second term of (11); thus the second term becomes identical to the first term, since $A \to -A$ as seen from (12), and we obtain the constrained integral

$$\overline{E}(\alpha) = 2\int_{-1/2}^{1/2} dx_1 \ldots \int_{-1/2}^{1/2} dx_N A\Theta(A). \qquad (13)$$

Before we proceed, consider the following integral:

$$J_k = \int_{-1/2}^{1/2} dt(t-a)^k \Theta(t-a). \qquad (14)$$

where k is a non-negative integer. The value of this integral depends critically on the value of a.

(i) If $a > \frac{1}{2}$ (i.e. the upper limit of the integral), the constraint $\Theta(t-a)$ is never satisfied for any value of t $(-\frac{1}{2} < t < \frac{1}{2})$ and the integral J_k vanishes. Hence we may as well write (14) as

$$J_k = \Theta(\frac{1}{2} - a) \int_{-1/2}^{1/2} dt(t-a)^k \Theta(t-a) \qquad (15)$$

(ii) If $-\frac{1}{2} < a < \frac{1}{2}$, equation (15) reduces to

$$\begin{aligned} J_k &= \Theta(\tfrac{1}{2} - a)\Theta(\tfrac{1}{2} + a) \textstyle\int_a^{1/2} dt(t-a)^k \\ &= \tfrac{1}{k+1}(\tfrac{1}{2} - a)^{k+1}\Theta(\tfrac{1}{2} - a)\Theta(\tfrac{1}{2} + a) \end{aligned} \qquad (16)$$

(iii) If $a < -\frac{1}{2}$, equation (15) reduces to

$$\begin{aligned} J_k &= \Theta(\tfrac{1}{2} - a)\Theta(-\tfrac{1}{2} - a) \textstyle\int_{-1/2}^{1/2} dt(t-a)^k \\ &= \tfrac{1}{k+1}[(\tfrac{1}{2} - a)^{k+1} - (-\tfrac{1}{2} - a)^{k+1}]\Theta(\tfrac{1}{2} - a)\Theta(-\tfrac{1}{2} - a). \end{aligned} \qquad (17)$$

The integral J_k of (15) is thus the sum of (16) and (17), that is

$$\begin{aligned} J_k = \tfrac{1}{k+1}\{&(\tfrac{1}{2} - a)^{k+1}[\Theta(\tfrac{1}{2} + a) + \Theta(-\tfrac{1}{2} - a)] \\ &-(-\tfrac{1}{2} - a)^{k+1}\Theta(-\tfrac{1}{2} - a)\}\Theta(\tfrac{1}{2} - a). \end{aligned} \qquad (18)$$

By using the identity (9) we reduce (18) to

$$J_k = \frac{1}{k+1}\{(\frac{1}{2} - a)^{k+1} - (-\frac{1}{2} - a)^{k+1}\Theta(-\frac{1}{2} - a)\}\Theta(\frac{1}{2} - a) \quad (19)$$

We further note that for the second term of (19) the constraint $\Theta(\frac{1}{2} - a)$ is already included in $\Theta(-\frac{1}{2} - a)$ and hence redundant. Therefore, we find the following result for the integral (14):

$$\begin{aligned} &\textstyle\int_{-1/2}^{1/2} dt(t-a)^k\Theta(t-a) \\ &= \tfrac{1}{k+1}\left[(\tfrac{1}{2} - a)^{k+1}\Theta(\tfrac{1}{2} - a) - (-\tfrac{1}{2} - a)^{k+1}\Theta(-\tfrac{1}{2} - a)\right]. \end{aligned} \qquad (20)$$

With repeated use of (20) we can now calculate the integral $\overline{E}(\alpha)$ given by (13).

To perform the x_1-integral we express $\overline{E}(\alpha)$ in the form

$$\overline{E}(\alpha) = 2\alpha_1 \int_{-1/2}^{1/2} dx_1 \int_{-1/2}^{1/2} dx_2 \ldots \int_{-1/2}^{1/2} dx_N (x_1 - a)\Theta(x_1 - a), \qquad (21)$$

with

$$a = (\alpha_2 x_2 + \cdots + \alpha_N x_N)/\alpha_1. \qquad (22)$$

Now, using (20) with $k = 1$ we do the x_1-integral and obtain

$$\begin{aligned} \overline{E}(\alpha) = &\tfrac{1}{2(2)}\tfrac{1}{\alpha_1} \textstyle\int_{-1/2}^{1/2} dx_2 \ldots \int_{-1/2}^{1/2} dx_N \\ &[+(+\alpha_1 + 2\alpha_2 x_2 + \cdots + 2\alpha_N x_N)^2 \\ &\quad \Theta(+\alpha_1 + 2\alpha_2 x_2 + \cdots + 2\alpha_N x_N) \\ &-(-\alpha_1 + 2\alpha_2 x_2 + \cdots + 2\alpha_N x_N)^2 \\ &\quad \Theta(-\alpha_1 + 2\alpha_2 x_2 + \cdots + 2\alpha_N x_N)] \end{aligned} \qquad (23)$$

The x_2-integral may be similarly performed. Note that both terms in (23) have the structure of (20) with $k = 2$, and each of them yields two terms when the x_2-integral is carried out. Hence, after the x_2-integral is done (23) becomes

$$\begin{aligned} \overline{E}(\alpha) = &\tfrac{1}{2^2(2.3)}\tfrac{1}{\alpha_1\alpha_2} \textstyle\int_{-1/2}^{1/2} dx_3 \ldots \int_{-1/2}^{1/2} dx_N \\ &[+(+\alpha_1 + \alpha_2 + 2\alpha_3 x_3 + \cdots + 2\alpha_N x_N)^3 \\ &\quad \Theta(+\alpha_1 + \alpha_2 + 2\alpha_3 x_3 + \cdots + 2\alpha_N x_N) \\ &-(-\alpha_1 + \alpha_2 + 2\alpha_3 x_3 + \cdots + 2\alpha_N x_N)^3 \\ &\quad \Theta(-\alpha_1 + \alpha_2 + 2\alpha_3 x_3 + \cdots + 2\alpha_N x_N) \\ &-(+\alpha_1 - \alpha_2 + 2\alpha_3 x_3 + \cdots + 2\alpha_N x_N)^3 \\ &\quad \Theta(+\alpha_1 - \alpha_2 + 2\alpha_3 x_3 + \cdots + 2\alpha_N x_N) \\ &+(-\alpha_1 - \alpha_2 + 2\alpha_3 x_3 + \cdots + 2\alpha_N x_N)^3 \\ &\quad \Theta(-\alpha_1 - \alpha_2 + 2\alpha_3 x_3 + \cdots + 2\alpha_N x_N)] \end{aligned} \qquad (24)$$

The remaining integrals, x_3 through x_N, may be carried out in a similar manner by using the integral (20) with $k = 3$ through $k = N$. There will be a proliferation of terms, for each time another variable is integrated out the number of terms doubles. Thus, we obtain 2^N terms altogether when all the integrals are done. The final result can be expressed in the compact form

$$\begin{aligned} \overline{E}(\alpha) = &\tfrac{1}{2^N(N+1)!}\tfrac{1}{\prod_{i=1}^N \alpha_i} \textstyle\sum_{\sigma_1=\pm1} \cdots \sum_{\sigma_N=\pm1} \\ &\left[(\textstyle\prod_{i=1}^N \sigma_i)(\sum_{i=1}^N \sigma_i\alpha_i)^{N+1}\Theta(\sum_{i=1}^N \sigma_i\alpha_i)\right] \end{aligned} \qquad (25)$$

Here we remark that in spite of the presence of the α_i's in the denominator of (25), this expression remains finite as any α_i goes to zero (as it should). For the sake of simplicity assume that $\alpha_N \to 0$. To verify that (25) does not diverge, we expand its numerator in powers of α_N and retain terms linear in α_N. Doing so we get

$$\begin{aligned} &\textstyle\sum_{\sigma_N=\pm1} \sigma_N(\sum_{i=1}^N \sigma_i\alpha_i)^{N+1}\Theta(\sum_{i=1}^N \sigma_i\alpha_i) \\ &\to 2(N+1)\alpha_N(\textstyle\sum_{i=1}^{N-1} \sigma_i\alpha_i)^N\Theta(\sum_{i=1}^{N-1} \sigma_i\alpha_i) \end{aligned} \qquad (26)$$

and thus (25) reduces to an identical expression with $N \to N-1$.

In Section 5, we present an efficient algorithm for computing the expression (25) for the average error.

4. ANALYTIC EXPRESSION FOR THE DISTRIBUTION OF ERROR

The minimum value that the error

$$E = |A| = |\alpha_1 x_1 + \cdots + \alpha_N x_N| \qquad (27)$$

can have is of course zero and the maximum value of the error is

$$E_{max} = \frac{1}{2}(\alpha_1 + \cdots + \alpha_N) \qquad (28)$$

corresponding to when all the variables have their maximum error $x_i = 1/2$ (or $-1/2$). Note that all the α_i are taken to be positive. In this section we calculate the probability for the error E falling in the interval $[\eta, \varsigma]$ where $0 < \eta < \varsigma < E_{max}$. We denote this probability by $P(\eta < E < \varsigma)$. The most convenient way of calculating this probability is to express it as

$$P(\eta < E < \varsigma) = P(E > \eta) - P(E > \varsigma) \tag{29}$$

and then calculate $P(E > \eta)$, i.e. the probability of the error E being larger than η. This probability, $P(E > \eta)$, is the sum of all the possible combinations of the errors x_i in individual variables that would yield a collective error E larger than a certain value η, which may be expressed as the constrained integral

$$P(E > \eta) = \int_{-1/2}^{1/2} dx_1 \ldots \int_{-1/2}^{1/2} dx_N \Theta(E - \eta). \tag{30}$$

We now calculate this integral for an arbitrary number of variables N and arbitrary values of parameters α_i and η.

By using $E = |A|$ from (27) and formula (10) we can write (30) as

$$P(E > \eta) = \textstyle\int_{-1/2}^{1/2} dx_1 \ldots \int_{-1/2}^{1/2} dx_N \\ [\Theta(A - \eta)\Theta(A) + \Theta(-A - \eta)\Theta(-A)]. \tag{31}$$

By making the transformation $x_1 \to -x_1, \ldots, x_N \to -x_N$ in the second term of (31), and noting that $A \to -A$, this term becomes identical to the first term and we get

$$P(E > \eta) = 2\int_{-1/2}^{1/2} dx_1 \ldots \int_{-1/2}^{1/2} dx_N \Theta(A - \eta)\Theta(A). \tag{32}$$

The constraint $A > 0$ can be eliminated since it is contained in $A > \eta > 0$ and is redundant; therefore

$$\begin{aligned} P(E > \eta) &= 2\textstyle\int_{-1/2}^{1/2} dx_1 \ldots \int_{-1/2}^{1/2} dx_N \Theta(A - \eta) \\ &= 2\textstyle\int_{-1/2}^{1/2} dx_1 \ldots \int_{-1/2}^{1/2} dx_N \Theta(\alpha_1 x_1 + \cdots + \alpha_N x_N - \eta) \end{aligned} \tag{33}$$

The above integral can be calculated in a manner similar to the average error integral, $\overline{E}(\alpha)$, with repeated use of the result (20). The x_1-integral can be performed by using (20) with $k = 0$, after which we obtain

$$\begin{aligned} P(E > \eta) = \tfrac{1}{\alpha_1} \textstyle\int_{-1/2}^{1/2} dx_2 \ldots \int_{-1/2}^{1/2} dx_N \\ [+(+\alpha_1 + 2\alpha_2 x_2 + \cdots + 2\alpha_N x_N - 2\eta) \\ \Theta(+\alpha_1 + 2\alpha_2 x_2 + \cdots + 2\alpha_N x_N - 2\eta) \\ -(-\alpha_1 + 2\alpha_2 x_2 + \cdots + 2\alpha_N x_N - 2\eta) \\ \Theta(-\alpha_1 + 2\alpha_2 x_2 + \cdots + 2\alpha_N x_N - 2\eta)]. \end{aligned} \tag{34}$$

Similarly the x_2-integral can be carried out by using (20) with $k = 1$, which yields

$$\begin{aligned} P(E > \eta) = \tfrac{1}{2(1.2)} \tfrac{1}{\alpha_1 \alpha_2} \textstyle\int_{-1/2}^{1/2} dx_3 \ldots \int_{-1/2}^{1/2} dx_N \\ [+(+\alpha_1 + \alpha_2 + 2\alpha_3 x_3 + \cdots + 2\alpha_N x_N - 2\eta)^2 \\ \Theta(+\alpha_1 + \alpha_2 + 2\alpha_3 x_3 + \cdots + 2\alpha_N x_N - 2\eta) \end{aligned}$$

$$\begin{aligned} -(-\alpha_1 + \alpha_2 + 2\alpha_3 x_3 + \cdots + 2\alpha_N x_N - 2\eta)^2 \\ \Theta(-\alpha_1 + \alpha_2 + 2\alpha_3 x_3 + \cdots + 2\alpha_N x_N - 2\eta) \\ -(+\alpha_1 - \alpha_2 + 2\alpha_3 x_3 + \cdots + 2\alpha_N x_N - 2\eta)^2 \\ \Theta(+\alpha_1 - \alpha_2 + 2\alpha_3 x_3 + \cdots + 2\alpha_N x_N - 2\eta) \\ +(-\alpha_1 - \alpha_2 + 2\alpha_3 x_3 + \cdots + 2\alpha_N x_N - 2\eta)^2 \\ \Theta(-\alpha_1 - \alpha_2 + 2\alpha_3 x_3 + \cdots + 2\alpha_N x_N - 2\eta)]. \end{aligned} \tag{35}$$

The remaining integrals, x_3 through x_N, can also be performed by using (20) with $k = 2$ through $k = N - 1$. The final result can be expressed in the compact form

$$P(E > \eta) = \tfrac{1}{2^{N-1}N!} \tfrac{1}{\prod_{i=1}^N \alpha_i} \textstyle\sum_{\sigma_1 = \pm 1} \cdots \sum_{\sigma_N = \pm 1} \\ \left[(\textstyle\prod_{i=1}^N \sigma_i)(-2\eta + \sum_{i=1}^N \sigma_i \alpha_i)^N \Theta(-2\eta + \sum_{i=1}^N \sigma_i \alpha_i).\right] \tag{36}$$

We make the following remark about the result just obtained: In spite of the appearance of α_i in the denominator of (36) this expression remains finite as any $\alpha_i \to 0$. This can be verified in a similar manner to (26), for as, e.g. α_N, goes to zero an identical expression is obtained with N replaced by $N - 1$.

5. AN ALGORITHM FOR THE COMPUTATION OF THE AVERAGE ERROR:

The formula (25) that we derived for the average error involves N summations and 2^N terms. Of these 2^N terms only *those terms contribute* whose corresponding Θ-functions have positive arguments. Other terms must be ignored. A simple algorithm capturing these points is as follows:

```
sum = 0.0
for i1=−1, step 2, 1
begin σ[1]=i1
  for i2=−1, step 2, 1
  begin σ[2]=i2
          .
          .
          .
          for iN=−1, step 2, 1
          begin σ[N]=iN
            sign = +1
            B = 0
            for i=1, step 1, N
            begin
              sign = sign * σ[i]
              B = B + σ[i]α[i]
            end
            if B > 0 then sum = sum + sign * $B^{N+1}$
          end
          .
          .
          .
  end
end
```

where $B = \sum_{i=1}^N \sigma_i \alpha_i$ is the argument of the Θ-function. Although for small and *fixed* values of N this algorithm may be convenient, when N is large or when it is a parameter to the program the algorithm exhibits several *shortcomings*: *i*) the length of the code becomes unduly long, as N loops are required; ii) as the number of loops in the algorithm depends on the value of N, the program itself has to be modified whenever N is changed; iii) the efficiency of the algorithm is far from optimum. The reason for the last shortcoming is that this algorithm computes the argument of the Θ-function, B, for all possible combinations of σ's, which amounts to 2^N times. As we shall see, it is possible to develop an algorithm that discards most combinations of σ's that yield negative B's, without even taking them into consideration.

In the algorithm we just described, at the first iteration σ_1 through σ_{N-1} remain unchanged, e.g. they remain -1, while σ_N undergoes its two possible states. In the second iteration σ_{N-1} changes its state and σ_N again undergoes its two states. This indicates that the number of σ's that are negative (and for that matter those that are positive) constantly changes. That is, for the four combinations mentioned above the number of positives were 0,1,1 and 2, respectively.

We now propose a different approach, namely one that keeps the number of negatives unchanged as long as possible. To this end, we will change the state of any σ as frequently as necessary. More precisely, after considering the configuration where all the σ's are positive, we consider configurations where all but one of the σ's is negative. (There are N such configurations.) Then we examine configurations having two negative σ's and so on. This allows us to have a short code, independent of the size of N.

To achieve our second goal which is the improvement of the efficiency of the algorithm, we first sort the α parameters so that they constitute a non-decreasing list. We call the smallest α the leftmost and the largest one the rightmost parameter in the list. The index of the smallest, i.e. the leftmost α, is 1, that of the one to its right is 2, and so on. After considering the configuration where all σ's are positive, we start with the configuration where only the leftmost σ (corresponding to the smallest α) is negative and then move this negative to the right. Now suppose that when this negative reaches position i on the list, B becomes negative. Since the α's are sorted, it is clear that the negative should not move any further to the right. This saves us the evaluation of $N-i$ configurations that have only one negative. After the configurations that have only one negative are exhausted (either because B has become negative or the negative has reached the rightmost parameter), we take the two leftmost parameters to be negative. Here, while the leftmost negative is kept at its spot, the other one is moved across the list until B becomes negative. At this point, one negative is temporarily fixed at the second spot while the other one, starting from the third spot, is moved to the right, and so on.

```
           Q                       R
 -  -  +  -  +    +    -  -  +  -  +  +
```

Figure 1

```
 -  -  +  +  -    -    -  -  +  +  +  +
```

Figure 2

The general strategy when a configuration with $B < 0$ is encountered is as follows. The closest negative σ, say σ_Q, to the rightmost negative σ, say σ_R, is found subject to the constraint that there are at least two positive σ's between R and Q. (R and Q are the positions of σ_R and σ_Q in the sorted list, respectively. See Figure 1.) If σ_Q exists, it is obvious that σ_{Q+1} is positive. Here, the negative at Q is moved to the position $Q+1$ and all the negatives to the right of the spot $Q+1$ are placed immediately next to it without any positive σ between them. See Figure 2. The reason for doing this is that when $B < 0$ is encountered, all possible configurations (subject to the constraint that all the negatives at Q and to its left are fixed) that give rise to $B > 0$ are exhausted. (This can be verified by close inspection of the string of positive and negative σ's in a sorted list.) Therefore, it is time to move the negative at Q forward and to evaluate the resulting configurations starting with the one where all the negatives to the right of $Q+1$ are positioned immediately next to it. Also, one can verify that when $B > 0$ but $R = N$, then there must be only one positive σ between R and Q. The rest of the steps are the same as in the case where $B < 0$ is encountered. Finally, if σ_Q does not exist, then it is time to increase the number of negatives. Or it may be that the evaluation of the integral is complete. Below we present a complete description of the algorithm.

If a sorting program is not available then one should use a different version of the algorithm. This is done in the Appendix.

6. THE ERROR DISTRIBUTION ALGORITHM

The expression (36) that we derived for the error distribution, $P(E > \eta)$, bears close resemblance to that for the average error. This implies: i) we need an algorithm for the numerical evaluation of our analytical result; ii) the algorithm can be derived from the one we proposed for the average error through minor modifications. The required changes are as follows:

Changes to the Definitions of the Parameters of the Algorithm:

The last item defined in the average error algorithm, namely E, should be deleted. Instead the following two items have to be added:

η: An input to the algorithm. The algorithm will compute the probability of the error to be greater than η.

P: The output of the algorithm, namely $P(E > \eta)$.

Changes to the Algorithm Itself:

In step (3) calculate B from $B = -2\eta + \sum_{i=1}^{N} \sigma_i \alpha_i$.

Line 11 in step (5) should read:

$c = 2^{N-1} N! \prod_{i=1}^{N} \alpha_i$. Set $P = sum/c$. Return.

Line 1 in step (6) should read:

Add the term $sign * B^N$ to sum.

Finally, we have to add one additional statement, namely

If $R = 0$ set $P = 0$, and return.

to the top of step (5).

7. EXAMPLES

Here we discuss several cases where the method can be applied to obtain information about the general reliability of vision computations.

7.1. Derivatives of the Intensity Field

We begin with the computation of the first difference of the intensity field. Let $f(x, y)$ denote the intensity field at pixel (x, y). The first difference of $f(x, y)$ in the x-direction, $f_x(x, y)$, is simply

The Average Error Algorithm:

Definitions:

N: The number of α parameters.

M: The number of σ's that are negative.

R: The rightmost σ whose value is -1.

$Q, N1$: When the situation is such that R should not move forward any further (either because of $B < 0$ or $B > 0$ and $R = N$), then the closest negative to R subject to the constraint that there are at least $N1$ positives between them has to be found. The place of this negative is called Q. (Note that if $B < 0$, $N1 = 2$, otherwise $N1 = 1$.)

PS: The number of positives between R and Q.

NG: The number of negatives between R and Q, including Q.

$flag$: is set to 1 when $R = N$ and $B > 0$. Otherwise, it is set to 0.

$sign$: $\Pi_{i=1}^{N}\sigma_i$. That is, $sign = 1$ if there is an even number of negative σ's and $sign = -1$ otherwise.

E: The output of the algorithm, namely the average error.

The Algorithm:

1) Sort the α's so that they constitute a non-decreasing list. The smallest α in the list is referred to as the leftmost element and the largest one as the rightmost element.
Also set $M = -1$, $sum = 0.0$, $g = -1$ and $flag = 0$.

2) Increase the number of negative σ's from M to $M + 1$, and place them at the $M + 1$ leftmost positions of the list. That is,
set $M = M + 1$, and
for $I = 1$ to $I = M$, set $\sigma[I] = -1$
for $I = M + 1$ to N, set $\sigma[I] = 1$.
Change the sign of g, i.e. set $g = -g$.
Also set $R = M$.

3) Calculate B from $B = \sum_{i=1}^{N}\sigma_i\alpha_i$.

4) If $B > 0$ and $flag = 0$ go to 6.

5) Here, either we have $B < 0$ or $flag = 1$. This implies that either the computation of the integral is complete or the configuration of positives and negatives needs extensive rearrangement:

```
    Set $flag$=0.
    If $B < 0$ set $N1 = 2$, else set $N1 = 1$.
    Set $NG = 1$.
    Start from $R$, go back to the left, increment $NG$ whenever a new negative
    is encountered.
    IF the beginning of the list is reached without $N1$ positives are encountered, THEN
        If $B < 0$ the computation of the integral is practically complete. Compute
            $c = 2^N(N+1)!\prod_{i=1}^{N}\alpha_i$. Set $E = sum/c$. Return.
        Else the number of negatives in the list must be increased. Go to 2.
    Else (i.e. when $N1$ positives are reached) set $PS = N1$ and go back
    further to the left. Increment $PS$ whenever a new positive is encountered.
        If the beginning of the list is reached before any negative is encountered,
            then the number of negatives must be increased. Go to 2.
        Else when the first negative is encountered, set $Q = R - NG - PS$.

    Rearrange the negatives:

        Move the negative at $Q$ to the position $Q + 1$ and place all the negatives
        to its right at the spots $Q + 2$ through $Q + NG + 1$.
        Set $R = Q + NG + 1$.
        Go to 3.
```

```
6) Add the term $sign * B^{N+1}$ to $sum$.
   If $R = 0$, go to 2.
   If $R < N$ move $R$ to the right. That is, set $\sigma[R] = 1$.
      Also, set $R = R + 1$ and $\sigma[R] = -1$.
      Calculate $B$ by setting $B = B + \alpha[R-1] - \alpha[R]$.
      Go to 4.
   Else set $flag = 1$. Go to 5.
```

$$f_x(x,y) = f(x,y) - f(x-1,y) \qquad (37)$$

(see, for example, Rosenfeld and Kak[5]). The range of the digitization error in either $f(x,y)$ or $f(x-1,y)$ is from $-1/2$ to $1/2$ units of gray level. The average value for the error in the computation of the first difference of the intensity field is then

$$\overline{E} = \int_{-1/2}^{1/2} dx_1 \int_{-1/2}^{1/2} dx_2 |x_1 - x_2| = \frac{1}{3} \qquad (38)$$

of the gray level unit.

In addition to computing the average error, it is interesting to know about the distribution of the error. This can be done by, say, plotting the error distribution using the tools already developed. Here, however, we content ourselves with a few samples. The maximum error is $E_{max} = 1$ gray level unit. The probability of the error being in the outer 75% range (i.e. being more that 3/4 gray level unit) is $P(E > 0.75) = 0.0625$, namely only 6.25%. Likewise, the probability of the error being in the outer 50% range (i.e. being more than 1/2 gray level unit) is $P(E > 0.5) = 0.25$.

The average value for the relative error in $f_x(x,y)$ is $\frac{1}{3|f(x,y)-f(x-1,y)|}$. By substituting the numerical value for the difference between $f(x,y)$ and $f(x-1,y)$ in the above expression one can obtain the average value of the relative error. The distribution of the relative error can be computed similarly.

Next we consider the second differences of the intensity field. For the second difference in the x-direction, f_{xx}, we have

$$f_{xx}(x,y) = f(x-1,y) + f(x+1,y) - 2f(x,y) \qquad (39)$$

with the average error

$$\overline{E} = \int_{-1/2}^{1/2} dx_1 \int_{-1/2}^{1/2} dx_2 \int_{-1/2}^{1/2} dx_3 |2x_1 - x_2 - x_3| = 0.5833 \quad (40)$$

The maximum error is $E_{max} = 2(1/2) + 1/2 + 1/2 = 2$ gray level units. The probability of the error being in the outer 75% range (i.e. being more that 1.5 gray level units) is $P(E > 1.5) = 0.0208$, namely only 2%. Likewise, the probability of the error being in the outer 50% range (i.e. being more than 1 gray level unit) is $P(E > 1) = 0.1667$.

For the second difference in the x and y directions, f_{xy}, we have

$$f_{xy} = f(x,y) + f(x-1,y-1) - f(x-1,y) - f(x,y-1) \quad (41)$$

with the average error

$$\overline{E} = \int_{-1/2}^{1/2} dx_1 \int_{-1/2}^{1/2} dx_2 \int_{-1/2}^{1/2} dx_3 \int_{-1/2}^{1/2} dx_4 |x_1 + x_2 - x_3 - x_4| = 0.4667 \qquad (42)$$

The maximum error is $E_{max} = 1/2 + 1/2 + 1/2 + 1/2 = 2$ gray level units. The probability of the error being in the outer 75% range (i.e. being more that 1.5 gray level units) is $P(E > 1.5) = 0.0052$, namely only 0.5%. Likewise, the probability of the error being in the outer 50% range (i.e. being more than 1 gray level unit) is $P(E > 1) = 0.0833$, i.e. 8.3%.

Note that the computation of f_{xx} (and similarly f_{yy}) is significantly less accurate than that of f_{xy}. This is in contrast to the fact that the maximum error for both differences is the same, namely, two gray level units. The numerical values of the relative error for the second difference f_{xx}, R_{xx}, can be obtained by substituting for the values of the f's in

$$R_{xx} = 0.5833/|2f(x,y) - f(x-1,y) - f(x+1,y)|. \qquad (43)$$

For the average value of the relative error not to be greater than, e.g. 10%, the second difference f_{xx} must be 6 units. Since frequently this is not the case, one may conclude that algorithms that involve the second differences of the intensity field are not reliable. The numerical value of the relative error for the second difference f_{xy} can be obtained from $0.4667/|f(x,y) + f(x-1,y-1) - f(x-1,y) - f(x,y-1)|$ for f_{xy}.

7.2. Computation of Rotation Angles in Stereo

The previous example was concerned with the quantization of the intensity field. Here, we consider examples that involve spatial sampling. An interesting example is the following. Kamgar-Parsi and Eastman [6] have shown that examples of sets of image points yielding a most stable computation of the relative roll angle in a stereo system with small relative angles are the combination of points shown in Figure 3 or that shown in Figure 4.

It is shown that for points (1,2,3) of Figure 3, the error in the roll angle, Δ_{roll}, is given by

$$\Delta_{roll} \simeq \frac{\Delta\delta_{1y} - \Delta\delta_{3y}}{2a} \qquad (44)$$

where a is half the width or length of the image planes.

The numerator is the error in the difference of the vertical disparities of points 1 and 3. (Note that the quantization error in point 2 does not contribute.) The maximum value for this expression is 2 pixels–one pixel error contained in each vertical disparity. Thus, the maximum error for the relative roll angle will be $1/a$ radians (note that a is now in pixels). For a typical camera we may assume that the focal length is $f = 7$ millimeters, the view angle is about 50 degrees and the image plane is 512 by 512 pixels. This implies that each pixel covers a square of 0.012 by 0.012 millimeters and half the width or length of the image plane, a, is about 3.1 millimeters. For such a camera we have $\Delta_{roll} = 0.22$ degrees.

Since $\Delta\delta_{1y} - \Delta\delta_{3y} = \Delta y_{1l} - \Delta y_{1r} - \Delta y_{3l} + \Delta y_{3r}$, the average error is given by

$$\overline{E} = \frac{1}{2a}\int_{-1/2}^{1/2}\int_{-1/2}^{1/2}\int_{-1/2}^{1/2}\int_{-1/2}^{1/2} |x_1 - x_2 - x_3 + x_4| dx_1 dx_2 dx_3 dx_4 \tag{45}$$

This quadruple integral is evaluated in (42). It is equal to 7/15. Therefore, on the average, for points (1,2,3) shown in Figure 3 the error in the roll angle is $7/30a$ radians or 0.05 degrees. Note that this error for most practical purposes is not too large.

For the configuration of image points shown in Figure 4 we have

$$\Delta_{roll} \simeq \frac{(f^2 + a^2)(\Delta\delta_{1y}/2 + \Delta\delta_{2y}/2 - \Delta\delta_{3y})}{2a(f^2 + a^2/2)} \tag{46}$$

The maximum error in the previous case is $1/a$ and in this case $(f^2+a^2)/a(f^2+a^2/2) > 1/a$ radians. Thus, the maximum error in the present case (which for our camera model turns out to be 0.24 degrees) is *somewhat larger than that of the previous case.* For the average error we have

$$\begin{aligned}\overline{E}_{roll} = \tfrac{f^2+a^2}{2a(f^2+a^2/2)} \textstyle\int_{-1/2}^{1/2} dx_1 \ldots \int_{-1/2}^{1/2} dx_6 \\ |0.5(x_1 + x_2 + x_3 + x_4) - x_5 - x_6|\end{aligned} \tag{47}$$

The above multiple integral is equal to 0.4043. Therefore, it can be shown that the average error in this case is $0.4404/2a$ radians. Note that this is less than the average error in the previous case, i.e. $.4667/2a$. The interesting point is that of the two cases that we considered one, namely *the latter, has a* greater maximum error but a smaller average error. The reason for this is that in the latter case three quantities, namely $\Delta\delta_{1y}$, $\Delta\delta_{2y}$ and $\Delta\delta_{3y}$ need to "enhance" each other in order to yield the maximum error. The previous case involved only $\Delta\delta_{1y}$ and $\Delta\delta_{3y}$ and, therefore, the probability of having maximum error was larger.

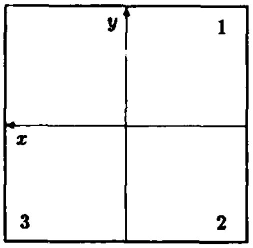


Figure 3

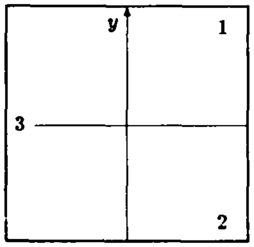


Figure 4

We now look at the probability of the error for the two cases being in the outer 75% and 50% ranges. For this we concentrate on the error distributions of $\Delta\delta_{1y} - \Delta\delta_{3y}$ and $\Delta\delta_{1y}/2 + \Delta\delta_{2y}/2 - \Delta\delta_{3y}$. The two expressions have the same maximum error, i.e. two pixels, but different error distributions. For the former expression we obtain $P(E > 1.5 \text{ pixels}) = 0.0052$ and $P(E > 1 \text{ pixel}) = 0.0833$, whereas for the latter we get $P(E > 1.5 \text{ pixels}) = 0.0007$ and $P(E > 1 \text{ pixel}) = 0.0417$. Note, however, that for the case of Figure 3, $P(E > 1.5 \text{ pixels})$ and $P(E > 1 \text{ pixel})$ correspond to $P(E > 0.165 \text{ degrees})$ and $P(E > 0.110 \text{ degrees})$, respectively, whereas for Figure 4 they correspond to $P(E > 0.18 \text{ degrees})$ and $P(E > 0.12 \text{ degrees})$. Therefore, to make a meaningful comparison between the two Figures, we calculate $P(E > 0.165 \text{ degrees})$ and $P(E > 0.110 \text{ degrees})$ for Figure 4 too. They turn out to be 0.0026 and 0.0644 respectively. *These errors do not seem to be very significant.* However, if we were dealing with points closer to the middle of the image plane the situation would be different. Suppose, for example, that we had the same configuration of points except that the distances of the points from the center of the image were all half what they are in Figures 3 and 4. In such situation for the two cases we would have the same error expressions, except that the value of a would be divided by two. Thus, for a case similar to Figure 3, we would have, e.g. $P(E > 0.20 \text{ degrees}) = 0.12$, namely 12%. We note that for the 3D recovery of object points *that are not close to the stereo system* 0.2 degrees error in the relative roll angle is quite significant. This is especially true because the error in the relative roll angle affects the error in the relative pan angle, and the error in the pan angle by itself increases quadruply when a decreases by half (see [6] and [7]).

7.3. Stereo Triangulation

As the final example we consider the problem of "Quantization Error in Stereo Triangulation" discussed by Blostein and Huang [8]. *The stereo system* that they have considered consists of two identical pinhole cameras. As regards the geometry of the stereo setup, they have assumed the two cameras are at equal heights and their image planes are on a common geometrical plane.

In addition to the above assumptions, they have implicitly assumed that the positions of the image centers are known precisely. Therefore, as far as the recovery of the 3-D position of a point in space is concerned the stereo triangulation reduces to a simple triangle, where the recovered projecting rays (joining the right and the left image positions of the point in space and the focal points) do intersect each other.

In what follows we employ the notations used in [8]. The origin of the 3-D world coordinate system is assumed to be midway between the two focal points. Let the X-axis point vertically upward, the Y-axis point towards the right camera and the Z-axis point straight ahead. Further, we assume that the pixels are squares of size dv. Let the coordinates of a point S on the object be (x, y, z) and the coordinates of its right and left images be (I_r, J_r) and (I_l, J_l) in pixel units, respectively. Ignoring the quantization error and using the stereo triangulation, the computed coordinates of point S will be

$$(x_c, y_c, z_c) = \left(\frac{-BI_l}{J_r - J_l}, B\left[\frac{1}{2} - \frac{J_r}{J_r - J_l}\right], \frac{Bf}{(J_r - J_l)dv}\right) \tag{48}$$

where B is the baseline, and $I_l = I_r$. The quantities I_l, I_r, J_l and J_r can be erroneous by up to half a pixel. Thus, denoting the exact (unquantized) coordinates of the image points by $(I_r + x_3, J_r + x_1)$ and $(I_l + x_3, J_l + x_2)$, we have $-1/2 \leq x_i \leq 1/2$, where i can be 1, 2 or 3. (Note that for this geometrically ideal stereo model the vertical quantization error is the same in both images.)

Like Blostein and Huang we compute the probabilities of the quantities $\varepsilon_z = \Delta z/z$, $\varepsilon_y = \Delta y/z$ and $\varepsilon_x = \Delta x/z$ being within a given range, where, e.g. Δz is the error in z due to quantization and ε_z is the scaled error.

7.3.1. Error in Depth

From equation (48) we obtain $\varepsilon_z = (x_1 - x_2)/D$, where $D = J_r - J_l$ is the observed disparity in pixels. An interesting analysis in [8] shows that when D is not too small, then we may assume that x_1 and x_2 are quantized independently, each of them following a uniform probability distribution. Hence, to compute the probability of $|\varepsilon_z|$ being smaller than, say τ, namely $P(|\varepsilon_z| < \tau)$ or $P(|x_1 - x_2| < D\tau)$ we use equation (36). By setting $N = 2$ and $\alpha_1 = \alpha_2 = 1$ we obtain

$$P(|\varepsilon_z| < \tau) = 1 - P(|\varepsilon_z| > \tau) =$$

$$1 - \tfrac{1}{4}[(-2D\tau + 2)^2\Theta(-2D\tau + 2)$$

$$+ \text{ terms having negative arguments to their } \Theta\text{-functions}] \tag{49}$$

$$P(|\varepsilon_z| < \tau) = \begin{cases} 1 & \text{if} \quad \tau \geq 1/D \\ 1 - (1 - D\tau)^2 & \text{if} \quad 0 \leq \tau < 1/D \end{cases} \tag{50}$$

The above result agrees with [8].

7.3.2. Error in the Horizontal Direction

Now we compute $P(|\varepsilon_y| < \tau)$. From equation (48) we obtain

$$R\varepsilon_y = \left(\frac{h}{D} - 1\right) x_1 - \frac{h}{D} x_2 \tag{51}$$

where $R \equiv f/dv$ and $h \equiv J_r$. As the above equation is written we have $\alpha_1 = |h/D - 1|$ and $\alpha_2 = |h/D|$. Depending on the values of h and D we need to distinguish several cases. In order to avoid possible confusion we introduce a new variable symmetrizing α_1 and α_2. Let $\gamma \equiv h/D - 1/2$. Now it is apparent that we need to distinguish only two cases, namely $|\gamma| < 1/2$ and $|\gamma| \geq 1/2$.

Case I): $|\gamma| < 1/2$

In this case $\alpha_1 = 1/2 - |\gamma|$ and $\alpha_2 = 1/2 + |\gamma|$. Note that we have arranged the two α_i's so that the smaller one is α_1. Using (36) we see that there are at most two terms that can have positive arguments in their Θ-functions. The results follow immediately:

$$P(|\varepsilon_y| < \tau) = \begin{cases} 1 & \text{if} \quad \mu \geq |1/2 \\ \frac{\mu - \mu^2 - \gamma^2}{1/4 - \gamma^2} & \text{if} \quad |\gamma| < \mu \leq 1/2 \\ \frac{2\mu}{1/2 + |\gamma|} & \text{if} \quad 0 \leq \mu < |\gamma| \end{cases} \tag{52}$$

where $\mu \equiv R\tau$.

Case II): $|\gamma| \geq 1/2$

In this case $\alpha_1 = |\gamma| - 1/2$ and $\alpha_2 = |\gamma| + 1/2$.

$$P(|\varepsilon_y| < \tau) = \begin{cases} 1 & \text{if} \quad \mu \geq |\gamma| \\ \frac{2|\gamma|\mu - \mu^2 - 1/4}{\gamma^2 - 1/4} & \text{if} \quad 1/2 \leq \mu < |\gamma| \\ \frac{2\mu}{1/2 + |\gamma|} & \text{if} \quad 0 \leq \mu < 1/2 \end{cases} \tag{53}$$

We note that in this case our results are in disagreement with those in [8].

7.3.3. Error in the Vertical Direction

Finally, we compute $P(|\varepsilon_x| < \tau)$. From equation (48) we obtain

$$\varepsilon_x = \frac{1}{R}[(v/D)(x_1 - x_2) - x_3] \tag{54}$$

where $v \equiv I_l$. Thus,

$$P(|\varepsilon_x| < \tau) = 1 - P(|\varepsilon_x| > \tau) = 1 - P(|\frac{v}{D}(x_1 - x_2) - x_3| > \mu) \tag{55}$$

where as before $\mu = R\tau$. From the above expression we see that $\alpha_1 = \alpha_2 = \nu$, where $\nu \equiv |v|/D$ (note that $D > 0$ always), and $\alpha_3 = 1$. By using equation (36) we have

$$P(|\nu x_1 + \nu x_2 + x_3| > \mu) =$$

$$(1/24\nu^2)[(-2\mu + 2\nu + 1)^3\Theta(-2\mu + 2\nu + 1)$$
$$-2(-2\mu + 1)^3\Theta(-2\mu + 1)$$

$$-(-2\mu + 2\nu - 1)^3\Theta(-2\mu + 2\nu - 1)$$
$$+(-2\mu - 2\nu + 1)^3\Theta(-2\mu - 2\nu + 1)$$

$$+ \text{ terms having negative arguments to their } \Theta\text{-functions}] \tag{56}$$

Hence, depending on the magnitude of ν we must distinguish three cases, $\nu > 1$, $1 \geq \nu > 1/2$ and $\nu \leq 1/2$. Denoting

$$C_1 \equiv \frac{(-2\mu + 2\nu + 1)^3}{24\nu^2}, \quad C_2 \equiv \frac{(-2\mu + 1)^3}{24\nu^2},$$
$$C_3 \equiv \frac{(-2\mu + 2\nu - 1)^3}{24\nu^2}, \quad C_4 \equiv \frac{(-2\mu - 2\nu + 1)^3}{24\nu^2}, \tag{57}$$

the results follow immediately:

CASE I): $\nu > 1$ or $|v| > D$

$$P(|\varepsilon_x| < \tau) = \begin{cases} 1 & \text{if} \quad 2\mu > 2\nu + 1 \\ 1 - C_1 & \text{if} \quad 2\nu - 1 < 2\mu \leq 2\nu + 1 \\ 1 - C_1 + C_3 & \text{if} \quad 1 < 2\mu \leq 2\nu - 1 \\ 1 - C_1 + 2C_2 + C_3 & \text{if} \quad 0 < 2\mu \leq 1 \end{cases} \tag{58}$$

CASE II): $1 \geq \nu > 1/2$ or $D \geq |v| > D/2$

$$P(|\varepsilon_x| < \tau) = \begin{cases} 1 & \text{if} \quad 2\mu > 2\nu + 1 \\ 1 - C_1 & \text{if} \quad 1 < 2\mu \leq 2\nu + 1 \\ 1 - C_1 + 2C_2 & \text{if} \quad 2\nu - 1 < 2\mu \leq 1 \\ 1 - C_1 + 2C_2 + C_3 & \text{if} \quad 0 < 2\mu \leq 2\nu - 1 \end{cases} \tag{59}$$

CASE III): $\nu \leq 1/2$ or $D/2 \geq |v|$

$$P(|\varepsilon_x| < \tau) = \begin{cases} 1 & \text{if} \quad 2\mu > 2\nu + 1 \\ 1 - C_1 & \text{if} \quad 1 < 2\mu \leq 2\nu + 1 \\ 1 - C_1 + 2C_2 & \text{if} \quad -2\nu + 1 < 2\mu \leq 1 \\ 1 - C_1 + 2C_2 - C_4 & \text{if} \quad 0 < 2\mu \leq -2\nu + 1 \end{cases} \tag{60}$$

Except for case I, the above results are in disagreement with those of [8]. Blostein and Huang report that in order to compute $P(|\epsilon_z| < r)$ for different cases they had to evaluate 26 (constrained) triple integrals by means of the mathematical software package MACSYMA. This may have led to algebraic mistakes. Recalling that in this example the number of quantized variables was only three, one may conclude that lacking a general formulation for the error distribution makes the problem highly unmanageable and time consuming for cases where the number of quantized variables is large.

8. CONCLUDING REMARKS

In computer vision only those approaches are of practical value that are capable of withstanding digitization errors, for no matter how accurately an experiment is done, digitization error will be present. Unfortunately, the computational approaches that are proposed in computer vision are not subject to any error analysis, or if they are, the sensitivity analysis is mostly superficial and of limited value. This may not be an unacceptable approach in other fields of science, but in computer vision where digitization error plays such an important role we need to have a careful error analysis of our computational approaches.

In this paper we have presented a method for genuine sensitivity analysis of algorithmic approaches. The method we have developed does not take into account errors due to experiments, but then such errors are not due to the intrinsic limitations of the approach in question. We have developed mathematical tools for the computation of the average error due to digitization and the probability of the error being within a certain range. The average error analysis is far more realistic and superior to the worst case analysis, because even if the maximum error is too great an algorithm may give acceptable results in most cases. The examination of the error distribution provides us with a powerful measure for assessing the applicability of an approach. That is, given the acceptable amount of error in the solution one can determine the probability of obtaining good results. Conversely, given the required confidence in the result one can calculate the amount of acceptable error.

The algorithms that we have presented for the computation of the average error and the error distribution of a given function of quantized variables are efficient and exact. They are also convenient to use.

Acknowledgements

We would like to thank Professor Azriel Rosenfeld for helpful comments and discussion.

References:

[1] Bevington, F.R., *Data Reduction* and *Error Analysis for the Physical Sciences*, McGraw-Hill, 1969.

[2] Beers, Y., *Introduction to the Theory of Error*, Addison-Wesley, 1957.

[3] Hogg, R.V. and Tanis, E.A., *Probability* and *Statistical Inference*, Macmillan, 1983.

[4] Brosh, M., private communication.

[5] Rosenfeld, A. and Kak, A.C., *Digital Picture Processing*, Academic Press, 1976.

[6] Kamgar-Parsi, B. and Eastman, R., "Calibration of a Stereo System with Small Relative Angles", CAR-TR-240, Center for Automation Research, University of Maryland, 1986.

[7] Kamgar-Parsi, B. "Minimization of the Quantization Error in Stereo", Proceedings of the DARPA Image Understanding Workshop, 1987.

[8] Blostein, S. and Huang T., "Quantization Errors in Stereo Triangulation", Proceedings of the First International Conference on Computer Vision, 1987.

Appendix

In this appendix we present a version of the Average Error Algorithm that does not require the coefficients (α_i's) to be sorted. This algorithm is not as efficient as the one given in the main text, but it is more convenient to use when a sorting program is not available. The definitions of the parameters are the same as those given in the algorithm presented in the main text.

```
1) Set $M = -1$, $sum = 0.0$, $g = -1$, and $flag = 0$.
2) Increase the number of negative $\sigma$'s from $M$ to $M + 1$, and place them
   at the $M + 1$ leftmost positions of the list. That is,
      set $M = M + 1$, and
      for $I = 1$ to $I = M$, set $\sigma[I] = -1$
      for $I = M + 1$ to $N$, set $\sigma[I] = 1$ .
   Change the sign of $g$, i.e. set $g = -g$.
   Also set $R = M$.

3) Calculate $B$ from $B = \sum_{i=1}^{N} \sigma_i \alpha_i$.
   If $B > 0$ add the term $sign * B^{N+1}$ to $sum$.
          If $R = 0$, go to 2.
          If $R < N$ move $R$ to the right. That is, set $\sigma[R] = 1$.
             Also, set $R = R + 1$ and $\sigma[R] = -1$. Go to 3.
          Else go to 4.

4) Here, $R = N$. This implies that either the computation of the integral is complete
   or the configuration of positives and negatives needs extensive rearrangement:
          Set $NG = 1$ and $PS = 1$.
          Start from $R$, go back to the left, increment $NG$ whenever a new negative
          is encountered.
          IF the beginning of the list is reached without one positive is encountered, THEN
                If there is no positive anywhere, the computation of the integral is practically complete.
                   Compute $c = 2^N (N + 1)! \prod_{i=1}^{N} \alpha_i$. Set $E = sum/c$. Return.
                Else the number of negatives in the list must be increased. Go to 2.
          Else (i.e. when a positive is reached) go back further to the left. Increment $PS$
          whenever a new positive is encountered.
               If the beginning of the list is reached before any negative is encountered,
                  then the number of negatives must be increased. Go to 2.
               Else when the first negative is encountered, set $Q = R - NG - PS$.

          Rearrange the negatives:

               Move the negative at $Q$ to position $Q + 1$ and place all the negatives
               to its right at positions $Q + 2$ through $Q + NG + 1$.
               Set $R = Q + NG + 1$.
               Go to 3.
```

Algebraic Reasoning in View Consistency and Parameterized Model Matching Problems *

David A. Cyrluk, Deepak Kapur[†]and Joseph L. Mundy

General Electric Company
Corporate Research and Development
Schenectady, NY 12345

Abstract

We proposed a view consistency approach in the 1987 DARPA Image Understanding workshop as a way to extract three dimensional information from multiple images of a polyhedral scene, without any knowledge of a model or information about the scene. We report further experiments in the use of this approach. We also explore the use of this approach for matching parameterized models in the absence of complete knowledge about the orientation of the camera. Towards this end, we have developed an algebraic reasoning system which combines a Gröbner basis algorithm for reasoning about polynomial *equational constraints* and *Bledsoe-Shostak-Brooks'* extended SUP-INF method for reasoning about inequality constraints. Inequalities can be used to express the range of parameter values as well as imprecise data. It is shown how the Gröbner basis algorithm extends the domain of application of the *SUP-INF* method as well as increases the efficiency of the SUP-INF method, especially in the presence of equational constraints. A number of improvements over Brooks' extension of the SUP-INF method are discussed.

1 Introduction

The use of explicit geometric models is believed to be an effective approach to recognize and locate objects in natural scenes [Besl and Jain; Brooks]. Geometric constraints imposed by a three-dimensional model can be used to eliminate false matches to background and clutter features. In addition, the process of matching the model to image data can produce the coordinate transformation between the model and the image reference frame. In many applications, numerically exact geometric models are often not available. Typically one may have some topological information about the model with partial knowledge of the geometry. Description of such models can be parameterized with parameters taking values within a certain range. There can also be situations in which no information about a model is available.

Many different methods have been proposed in the literature to generate three dimensional models; these include building models from CAD tools [Brooks], line drawings [Start] , range data [Connolly and Stenstrom], and stereo data.

The goal of this research effort is to perform object matching in images when numerically exact information about the model is not available. The object matching process can rely only on parameterized models or, in the worst case, use one image as a model to match against another image. In the last Image Understanding Workshop in 1987, we discussed an approach towards addressing this problem. This approach involves the use of geometric and algebraic reasoning methods to generate a set of constraints on the geometric and topological structure of an object from its image. These constraints are used as a model for matching against another image. We have developed and experimented with techniques for reasoning about geometry relationships. A geometric and algebraic reasoning system, *GEOMETER*, has been developed.

In [Cyrluk et al], we reported on preliminary experiments about the use of this approach for simple ideal images. Since then, the approach has been further developed, and improved by precomputing certain algebraic relationships as well as by making improvements in the algebraic reasoning algorithms used in *GEOMETER*. We have been able to improve the performance by an order of magnitude and in some cases, by two orders of magnitude. The improved method has been successfully used to extract three dimensional information from a more complex polyhedral object.

In our attempts to deal with imprecisely specified coordinate positions of image points as well as errors in data, *GEOMETER* has been extended to manipulate inequality constraints. This extension also provides the ability to specify parameterized models which can be used to match against images. In a parameterized model, certain geometric information of a model is incompletely specified using parameters taking values within a certain range.

In the next section, we review the view consistency approach. We discuss the algebraic formulation of this problem. We give an overview of the Gröbner basis approach for determining the consistency of nonlinear equational algebraic constraints. For details of the method, the reader may refer to our paper in the 1987 DARPA Image Understanding proceedings [Cyrluk et al]. We report on additional experiments for determining 3-dimensional structure from multiple two-dimensional images of polyhedral blocks.

*The work reported here was partially supported by the DARPA Strategic Computing Program under the Army Engineer Topographic Laboratories, Contract No DACA76-86-C-0007

[†]Current address: Department of Computer Science, State University of New York at Albany, Albany, NY 12222

In Section 3, we briefly discuss the SUP-INF method for reasoning about linear inequalities as proposed by Bledsoe [1975] and improved by Shostak [1977]. Then, we overview Brooks' extension [Brooks] to the method for handling a subclass of nonlinear inequality constraints. This is followed by a discussion of our implementation of Brooks' method in GEOMETER and its integration with the Gröbner basis method. A number of observations and improvements are discussed which extend the applicability of the SUP-INF method as well as improve its performance by processing equality constraints using the Gröbner basis method. An extension to the combined method is discussed in which variables are split based on knowledge of their lower and upper bounds; this extension produces better intervals of values that variables can have when constraints are satisfiable.

Subsequently, a number of experiments on a very simple example from image understanding application are discussed. In this example, image coordinates have errors, and a model is parameterized.

2 View consistency approach

The view consistency approach [Cyrluk et al] is based on an algebraic method to extract a partial model from a single two-dimensional perspective view of an object and the use of this model to recognize the object from another viewpoint. The *view consistency problem* is defined as given a two-dimensional image of a scene, determine whether another two-dimensional image could possibly be of the same scene or not. It is not possible to derive a complete, three-dimensional model from a single view without assuming a great deal about the relationships between object surfaces and edges. For example a common assumption is that the object is a polyhedron with adjacent faces and edges perpendicular [Herman]. With this assumption, it is possible to establish the three-dimensional structure of the visible surfaces from a single view. Using multiple images of a scene, it is possible to derive three dimensional information such as depth as well as information about the orientation and position of the camera if that is not known.

In our investigation, we consider polyhedron objects and assume that the perspective image transformation can be approximated by an affine transformation. An affine transformation is an orthographic projection with subsequent scaling of coordinate axes. An affine approximation is more realistic than the usual orthographic assumption, but is more constraining than the full perspective case. We however think that perspective transformation can also be handled in our approach.

2.1 Algebraic formulation of the problem

The least restrictive constraint that can be derived from the projection is simply that the image vertices are related to the corresponding object vertices by projection equations where the three-dimensional coordinates of each vertex are unknown, along with the six affine projection parameters.

Once a set of projection constraints is established from a given view of the object, they can be used as a model for recognition. The projection constraints are expressed as a set of algebraic equations in terms of unknown three-dimensional coordinates including depths of object vertices and transformation parameters. These depths cannot be determined from a single two-dimensional view. If we consider another two-dimensional projection, which is hypothesised to be another view of the object, then the equations derived from each projection should be algebraically consistent; of course, the correct assignment has to be made between the corresponding vertices of each view. Instead of assigning a single vertex in one image to a vertex in another image, a group of vertices in one image are assigned to a corresponding group of vertices in the other image. One example of such a group is to identify a cycle of edges and vertices in the projection image that corresponds to a visible face of the three-dimensional polyhedron; labeling can be helpful in determining such cycles [Cyrluk et al; Barry et al]. The fact that the cycle elements must be coplanar can then be used to further constrain the three-dimensional structure.

The process of recognition is to first form assignments between vertex groups in each projection and then to test the algebraic consistency of the resulting equations using the coplanarity constraints on vertex groups. If the equations are consistent, then the two views may correspond to the same three-dimensional object; or at least they share some solutions for the possible objects that are consistent with both projections. The consistency also allows a more specific determination of the three-dimensional configuration of the object. For example, if the transformation between the views is given in advance, then the correspondence between equations is similar to the feature matching done in classical stereo analysis. The matching between vertices provides the relative depth value for each vertex match.

If the transformation between views is unknown, then the depths cannot be determined, but only constrained. The object surface depth is not explicitly determined but the solutions of the equations represent a space of possible object surfaces. The introduction of new constraints, either from hypotheses about geometric constraints on groups of projection elements, or from new views of the object, will reduce the number of unknown coordinate values. In this sense, the approach is incremental.

If an inconsistency is detected, however, then other assignments of vertex groups have to be tried. Various heuristics such as the number of vertices in a vertex group and adjacency information about vertices in a vertex group can be used to reduce the search space of possible assignments between the vertex groups of the two images.

In summary, the following are the key steps in our method:

1. Segment images and identify faces (using labeling).

2. Match a face in the first image to a face in the second image, i.e., assign vertices on the face in the first image to vertices on the face in the second image.

3. Check for consistency:

   (a) If the match is way off, an inconsistency will usually be detected in the first match itself. If all vertex assignments between the two faces are inconsistent, then backtrack and try to match the first face against some other face in the second image. If no such face is left in the second image, then the two images are not of the same object.

   (b) Otherwise, use the three dimensional information already derived in the form of algebraic relations to match a second face in the first image against another face in the second image.

(c) Keep matching faces in one image against the corresponding faces in the other image until the desired three dimensional information is derived.

The use of feasible constraints for object recognition is motivated by the *ACRONYM* system [Brooks]; see also [Sugihara]. In *ACRONYM*, the model of an object is known; the consistency between a known three-dimensional model and its projection in an image was tested using an extension of the SUP-INF method of reasoning about linear inequalities. The inequalities arise because of tolerances on the expected transformation and image feature positions. By contrast, in the work reported here, we do not have a three-dimensional model, but only the constraints imposed by a two-dimensional view.

2.2 Testing Consistency using the Gröbner basis approach

The consistency of the two views can be established by deciding whether algebraic constraints corresponding to the two views are consistent. These algebraic constraints are typically nonlinear. In [Cyrluk et al], we assumed that these constraints are non-linear equations, which is the case for ideal images in which image coordinates are exactly known. The experiments were able to demonstrate the feasibility of this approach in such cases. A key result is that relatively few constraints are sufficient to prove the inconsistency between two views. In the case that the views are consistent, it is then possible to determine the transformation between views and extend the model to include explicit three-dimensional constraints. Later, in this paper, we use non-linear inequality constraints also.

There are four types of algebraic constraints obtained by matching vertex groups in one image to vertex groups in the other image:

1. Equations describing the affine projection relating object points in terms of image points and projection parameters.

2. Geometric constraints describing relationships in each vertex group. These equational constraints restrict vertices in each vertex group to be coplanar.

3. Equations describing hypotheses about possible assignments between vertices in one image to vertices in the other image.

4. Trigonometric identities about the rotational angles in an affine transformation.

In addition, if the relation between two affine projections is known such as in the case of stereo matching, then there are additional equational constraints relating them also. The details about how such equations are generated are discussed in [Cyrluk et al].

Given these equational constraint, we must be able to solve the following problems:

1. Is a given assignment of vertex groups in two projections consistent?

2. Given a consistent assignment, what can we conclude about the transformation parameters and depth of the object surfaces?

3. Given two vertex groups with consistent assignments, can they be merged in a consistent manner?

A Gröbner basis algorithm [Buchberger, 1965; 1985] is used to check whether these equations are consistent or not. In case the system of equations is consistent, their Gröbner basis embodies all the information about the model which can be extracted from images. In this sense, computing a Gröbner basis serves the role of compilation of available knowledge. This information is stored for subsequent manipulation when equations corresponding to additional constraints are introduced.

The concept of a Gröbner basis of a finite set of equations was introduced by Buchberger [1965; 1970] in which he also gave an algorithm for computing such bases. Gröbner basis computation can be used to determine solutions of a system of nonlinear polynomial equations. In particular it can determine the consistency of a system of polynomial equations. For details of this application of a Gröbner basis algorithm, the reader may consult [Buchberger, 1985; Kapur; Cyrluk et al]. Below, we give the key results used and an overview of the approach.

Let Q be the field of rationals. Let $Q[x_1, ..., x_n]$ be the set of all polynomials with indeterminates x_1, ..., x_n. Let $p_1 = 0$, ..., $p_j = 0$ be a finite system of nonlinear equations in which p_1, ..., p_j are polynomials in $Q[x_1, ..., x_n]$.

Proposition: $p_1 = 0$, ..., $p_j = 0$ are not consistent, i.e. do not have a solution in complex numbers, if and only if a Gröbner basis of $p_1, ..., p_j$ includes 1.

Proposition: The set of all solutions in complex numbers of $p_1 = 0$, ..., $p_j = 0$ are also the common zeros of all polynomials of a Gröbner basis of p_1, ..., p_j , and vice versa.

The main advantage of computing a Gröbner basis of the set of polynomials p_1, ..., p_j corresponding to a consistent set of equations $p_1 = 0, ..., p_j = 0$ is that using the Gröbner basis, all solutions of the equations can be systematically examined; see [Buchberger, 1985] for details.

Polynomials in this approach are viewed as rewrite rules which can be used to generate normal forms for polynomials with respect to equational constraints. A Gröbner basis algorithm is similar to the Knuth-Bendix completion procedure in which additional polynomials are generated from a given set of polynomials until it is possible to generate canonical forms (unique normal forms) for polynomials with respect to equational constraints. The key inference step (apart from rewriting) is to generate a *critical pair* from two polynomials as illustrated by the following example:

Given two polynomials $x^2y - 3$ and $xy^2 - 1$, the least common multiple of their head-monomials can be computed, which is called *superposition* or *overlap*. In this case, the superposition is x^2y^2. Now, either of the above two polynomials can be used to rewrite the superposition. Rewriting with the first one gives $3y$ as the result, whereas rewriting with the second one produces x. The pair $<3y, x>$ is called the *critical pair* of the above polynomials. The equation $3y - x = 0$ follows from the equations $x^2y - 3 = 0$ and $xy^2 - 1 = 0$. A critical pair is further rewritten until no more rewriting is possible. If normal forms of the two polynomials in a critical pair are not the same, then a new polynomial is obtained by taking the difference of the two normal forms; this new polynomial is augmented to the original set. This process of generating new polynomials is continued until no new polynomials can be generated. This process is guaranteed to terminate because of Hilbert's Basis Condition, a combinatorial property of ideals over Noetherian rings [van der Waerden]. If the polynomial 1 (in fact, any rational number) is generated in this process, this indicates the original set of equations is inconsistent; otherwise, the final set of polynomials is a Gröbner basis of the input set of polynomials.

For further details, the reader may consult [Buchberger, 1985; Cyrluk et al].

2.3 Experiments

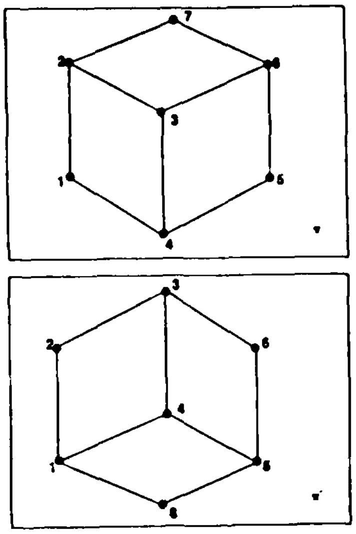
As reported in [Cyrluk et al], an implementation of a Gröbner basis algorithm was developed as a geometric reasoning system *GEOMETER* to prove geometry theorems [Kapur]. This implementation was subsequently modified for the view consistency approach. In [Cyrluk et al], we reported on preliminary experiments illustrating the view consistency approach on a cube, both when the relationship between projections is known as well as when the relationship between projections is not known. Since then, we have made many changes to our implementation, in particular incorporating the optimizations suggested in [Buchberger, 1979] as well as in [Kapur, Musser and Narendran, 1985] that reduce the number of critical pairs that have to be considered. We also modified the completion procedure to be patterned after a Knuth-Bendix completion procedure as implemented in *RRL*, a rewrite rule laboratory, a theorem prover based on rewriting techniques [Kapur, Sivakumar and Zhang].

Another key modification that helped improve the performance was to precompute some commonly occurring equational constraints that can be derived from the properties of the affine projections. As the following table contrasting the current results from the results reported in the 1987 workshop illustrates, these changes led to significant improvements in the performance (Figure 1). All experiments were done on a Symbolics 3600 LISP machine. The following table gives the timings for the case when the relation between the projections is known.

Problem	**At 1987 Workshop**	**Currently**
Depth	272 sec	20 - 30 sec
Inconsistency	48 sec	10 sec

The following table gives the timings for the case when the relation between the projections is not known.

Problem	**At 1987 Workshop**	**Currently**
Assigning one face in one image to another face in another image	1 hour	3-4 minutes
Inconsistent extended assignment	> 4 hours	7-10 minutes
Consistent extended assignment	> 10 hours	20 sec -5 minutes



π: x-axis: 45, y-axis: -35.26, z-axis: -30

π': x-axis: -45, y-axis: -35.26, z-axis: 30

Figure 1: Two projections of a Cube.

We recently tried the view consistency approach on a more complex polyhedral object (Figure 2). The results were extremely encouraging when the relation between projections is known. Our implementation could deduce depth parameters in less than a minute. However, when the relation between the projections are not known, although the method was still able to deduce depth parameters as well as the transformation parameters, it took considerably longer.

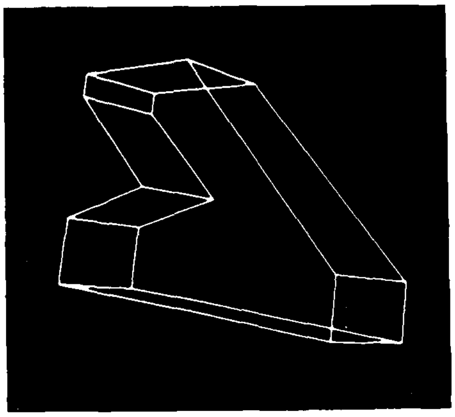


Figure 2: Block.

3 Reasoning about Inequalities

Two serious limitations of the use of the Gröbner basis algorithm for the view consistency problem are (i) the requirement that all the information should be specified in exact form, and (ii) the inability to deal with inequalities which can be used to express inexact data. For instance, if a coordinate position in an ideal projection is $\sqrt{20}$, this must be specified implicitly using an equation $x^2 = 20$; further, it is not possible to rule out the negative value that x could take. However, the value of x could be given to any desired precision using inequalities. Inequalities can also be used to describe parameterized models as discussed later; see also [Brooks].

A number of techniques have been proposed in the literature for reasoning about inequalities. In the case of linear inequalities, the well-known SIMPLEX algorithm for solving linear programming problems, and polynomial time algorithms of Kharhiyan and Karmarker can be used for large problems. However, for medium-sized problems, especially arising in program verification application, the SUP-INF method proposed by Bledsoe [1975] and extended by Shostak [1977] have been more widely cited; see also [Shostak, 1981]. An advantage of the SUP-INF method is that it can be easily implemented without much overhead.

Any decision procedure for the theory of real closed fields (also known as the theory of elementary algebra and geometry), such as Tarski's method, is a complete decision procedure for determining the inconsistency of nonlinear inequalities over the reals. There exist many complete methods for the theory of real closed fields. The method of cylindrical algebraic decomposition due to Collins has been implemented as a part of the SAC-2 system [Arnon et al]. Unfortunately, this implementation needs considerable computational resources for solving even simple problems.

The SUP-INF method was extended by Brooks in a straightforward way to consider non-linear inequalities and trigonometric functions. The extended method is not complete; actually, the subclass of non-linear inequalities for which the method works is not even known. Brooks used the extended method in the *ACRONYM* system for model-based recognition. We have implemented this algorithm in *GEOMETER* and experimented with it on a number of simple examples; we left out the trignometric component of the algorithm since we treat $sin(\theta)$ and $cos(\theta)$ as variables instead of functions of θ. We relate them with the equation $sin^2(\theta) + cos^2(\theta) = 1$.

Our implementation integrates Brooks' SUP-INF method with the Gröbner basis algorithm so that equational constraints are solved using the Gröbner basis algorithm and inequality constraints are manipulated using Brooks' algorithm. This integration of the Gröbner basis algorithm with Brooks' SUP-INF method will, henceforth, be referred to as the *combined method*. The combined method suffers from deficiencies similar to those of the SUP-INF method. Below, we discuss a number of improvements made to the combined method and show that the combined method not only works better than Brooks' SUP-INF method but also extends the domain of applicability of the SUP-INF method.

First, we give an overview of the original SUP-INF method for linear inequalities, and then discuss Brooks' extension for the nonlinear case. This is followed by a discussion of our improvements and extensions.

3.1. SUP-INF method

Bledsoe introduced the SUP-INF method to prove universally quantified formula in the theory of Pressburger arithmetic [Bledsoe]. This method was subsequently extended by Shostak [1977].

The input to the method is a conjunction of linear inequalities with rational coefficients. An equality constraint $p_1 = 0$ in the method is transformed into a conjunction of two inequalities $p_1 \leq 0$ and $p_1 \geq 0$. The goal is to decide whether these inequalities can be satisfied. If yes, the method produces lower and upper bounds for each variable appearing in the set of inequalities. A set of inequalities are unsatisfiable if and only if there is a variable for which its lower bound is greater than its upper bound.

The method is based on transforming inequalities such that for each variable x, they can be expressed as $x \leq ub_i$ or $x \geq lb_i$, where ub_i and lb_i are, in general, linear expressions in terms of the rest of the variables. An upper bound for x, SUP(x), is then the minimum over ub_i's, where as a lower bound for x, INF(x), is the maximum over lb_i. To compute the minimum of ub_i's, say, the algorithm is called recursively on each ub_i in an attempt to compute the lower and upper bounds on ub_i in terms of the variable x. Finally, linear equations in terms of x are obtained for these bounds, which can be solved. A dual technique is used for computing the lower bounds. In this way, the algorithm computes rational upper and lower bounds for each variable.

Shostak proved that these bounds are indeed tight. In order to decide whether a given set of inequalities have an integer solution, Shostak suggested further improvements to Bledsoe's method. If the inequalities do not have a real solution, then they do not have an integer solution either. However, if they do have a real solution, i.e., the method produces satisfiable intervals of upper and lower bounds for each variable, it can be checked whether each interval has an integer. If for some variable, its interval does not include an integer, the inequalities do not have an integer solution. Otherwise, the procedure has to search over integers in the satisfiable interval of each variable. The improvements reported below to the SUP-INF method and the combined method should also work for integer solutions.

3.2 Brooks' extension

Brooks extended Bledsoe-Shostak's SUP-INF method to be applicable to nonlinear inequalities including trigonometric functions. For handling nonlinear terms in an expression, the extended algorithm attempts to determine the parity (whether the value is always positive, zero or negative) of variables. Parity constraints from nonlinear terms are propagated to generate parity constraints on variables, which in turn, constrain the parity of other nonlinear terms involving these variables. For instance, if the parity of xy is known to be positive, then x and y will have the same parity, either both are positive or both are negative. The method is limited, however, since for computing lower and upper bounds of nonlinear expressions, it deals with each nonlinear term separately without constraining other terms in which common variables appear. For instance, as also pointed out by [Sacks], for computing an upper bound of $x^2 - x$ when x is unconstrained, an upper bound for x^2 and a lower bound for x are computed independently and the same value of x is not used for determining bounds of $x^2 - x$.

Despite these limitations, Brooks found this extension quite adequate for his work in model-based recognition.

4 Combining Gröbner basis algorithm and SUP-INF method

We have implemented Brooks' extended SUP-INF method in *GEOMETER* and integrated with the Gröbner basis algorithm. The integration is not completely done yet, as some of the steps are currently performed manually. The basic idea is to first manipulate equality constraints using the Gröbner basis algorithm, possibly deducing additional equality constraints. Equality constraints are then used as rewrite rules to simplify the inequality constraints. All constraints (input as well as deduced) are then transformed into inequalities and the SUP-INF method is invoked. If any new equality constraint is detected (when the satisfiable interval for some variable or term includes only one value, i.e., its upper bound and lower bound are identical), then that equality constraint is further propagated using rewriting and the critical pair computation in the Gröbner basis algorithm. Note that this method will only determine whether a variable or a term is equal to some rational value, and it will not determine whether a nonlinear expression is equal to a rational value.

Another modification made to Brook's extended sup-inf algorithm is to specify a subset of the variables as input variables. When SUP and INF are called on these variables, no extra computation takes place, rather the user provided bounds are used. This modification is especially useful in our parameterized model matching problems where there are a lot of variables involved. Below, we discuss some additional extensions to the SUP-INF method in our integration which makes it more effective.

4.1 Explicit use of equality constraints

As stated earlier, the SUP-INF method transforms equality constraints into equivalent inequality constraints. As a result, it does not make direct use of equality constraints for simplifying inequality constraints. As illustrated by the following example, using equality constraints to simplify inequality constraints considerably improves the performance of the method.

Given a set of linear constraints,

$3x_1 - 2x_2 - 7x_3 = 0$

$10x_3 - 8x_1 - 7x_2 = 0$

$5x_3 \geq (2x_2 + 3)$

$x_3 \geq (x_1 + x_2)$

$5 \geq 3x_1$

$3x_2 \geq -(x_1 + 4x_4)$

$2x_3 \geq 5x_4$

$4x_5 \geq -(x_2 + x_1)$

$2x_3 \geq 7x_5$

The SUP-INF method takes nearly 8.2 seconds to compute the satisfiable intervals:

$0.87341774 \leq x_1 \leq 1.6666666,$

$-0.62801933 \leq x_2 \leq -0.32911393,$

$0.46835443 \leq x_3 \leq 0.8937198,$

$0.028481012 \leq x_4 \leq 0.35748792,$

$-0.25966182 \leq x_5 \leq 0.2553485$

If the equality constraints are manipulated using the Gröbner basis algorithm (or any other method for solving linear equations such as Gaussian elimination) and the normalized equalities are used to rewrite the inequality constraints, we get the result in the total time of less than 1 second. Using the Gröbner basis algorithm, from the equality constraints, it is first determined that

$x_1 = -69/37x_3$ *and* $x_2 = 26/37x_3$.

These two equalities are used to eliminate x_1 and x_2 from inequality constraints. The improvement in performance is obtained because the SUP-INF method has to consider less number of variables in this case.

We have recently come to know that Käufl [1986] made similar observations about the SUP-INF method while investigating its use in program verification and proposed the use of equality constraints as rewrite rules for eliminating variables and simplifying inequality constraints. Käufl proposed a technique for determining additional linear equalities from linear inequalities by introducing new variables; we have not investigated the effectiveness of his method. However, it is not clear how this method will help in the presence of nonlinear inequalities.

The SUP-INF method produces the same results no matter in what order the bounds for variables are computed. However, it does seem that the order in which the bounds are computed affects the computational performance quite a bit. The method seems to implicitly consider all possible orderings on variables to compute bounds; this happens when each variable in an inequality is solved by moving the rest of the expression in the inequality onto the other side. For example, an inequality constraint $x_1 + x_2 - x_3 \geq 0$ is reformulated as $x_1 \geq (-x_2 + x_3)$ for computing bounds on x_1, as $x_2 \geq (-x_1 + x_3)$ for computing bounds on x_2, and $x_3 \leq (x_1 + x_2)$ for computing bounds on x_3.

4.2 Storing intermediate results

As observed by Brooks, the extended SUP-INF method is highly recursive. We observed while running several examples that many computations of upper and lower bounds of variables expressed using a given set of variables (the result of SUPP and INFF) were being repeated. As a result, we decided to store intermediate results; for each variable or a term, we store its upper-bound and lower-bound as expressions in terms of a given set of variables. Thus, whenever bounds on variables need to be computed, it is first checked whether bounds expressed in a given set of variables are already available; if so, they are retrieved. Only if they are not available, they are computed. It is possible to further improve this feature. For example, if a bound of a variable in terms of a set of variables is needed, it is sufficient to start with its bound in terms of a superset if that is available, and use that bound to compute a better bound in terms of the required subset. We have so far implemented the simplest memory feature.

For a modified version of the example discussed in the previous section in which the first two equality constraints were modified to inequality constraints by changing = to ≥, the SUP-INF method takes over 16 seconds without the memory feature. With the memory feature, however, only 4 seconds were needed to compute the bounds. Similar improvements in performance were observed on other examples. The improvement in performance in fact appears to increase nonlinearly with the size of an example; the improvement is dramatic for larger examples. For the second example discussed in the section below on imprecise data and parameterized model, the SUP-INF method with the memory feature took 20 minutes where as the version without the memory feature did not finish overnight.

4.3 Multiple runs of SUP-INF method for nonlinear inequalities

In the case of nonlinear inequality constraints, the SUP-INF method relies heavily on determining the parity of variables. The order in which variables are processed becomes important because, while determining bounds on a variable x, parity of some other variables may not be known. However, those parities may be determined subsequently from which it may be possible to obtain better bounds on x. In order to get those better bounds, there may be a need to run the SUP-INF method more than once. Examples can be constructed in which the SUP-INF method may have to be run the number of times proportional to the number of variables in order to obtain best possible bounds in the presence of nonlinear constraints.

The first example illustrates that two runs are needed to get the best bounds. The next example illustrates the need for three runs to get the best bounds. Variable x is processed first, followed by y and then z.

$x \geq 2, \quad y \geq 1, \quad z \leq 9,$
$2z \geq 3x \quad , yz \leq 6$

After the first run, the SUP-INF method returns the following bounds:

$2 \leq x \leq 4, \quad 1 \leq y \leq \infty, \quad 3 \leq z \leq 6.$

While computing an upper bound for y, it does not know the parity of z. However, after the parity of z is known, as is the case in the second run, a better upper bound for y is computed. The result after the second run is:

$2 \leq x \leq 4, \quad 1 \leq y \leq 2, \quad 3 \leq z \leq 6.$

Similarly, for the following example in which the variables are processed in the order, x first, y, z and lastly u, three runs are needed to get the best bounds:

$x \geq 12, \qquad y \geq 1, \qquad z \geq 0, \qquad u \leq 3,$
$2zu \geq 3x, \qquad yzu \leq 20, \quad (yz + zu) \geq 12$
$(yz + zu) \leq 40$

After the first run, the following bounds are obtained:

$12 \leq x \leq 44, \quad 1 \leq y \leq \infty,$
$6 \leq z \leq \infty, \quad 0 \leq u \leq 3.$

After the second run, the bounds are

$12 \leq x \leq 40/3, \quad 1 \leq y \leq 10/9,$
$6 \leq z \leq 40, \quad 9/11 \leq u \leq 3.$

And, finally the third run gives the best bounds,

$12 \leq x \leq 40/3, \quad 1 \leq y \leq 10/9,$
$6 \leq z \leq 220/9, \quad 9/11 \leq u \leq 3.$

This example also indicates that the order in which variables are processed, becomes extremely important if we wish to reduce the number of runs to obtain the best bounds.

Even for an unsatisfiable set of nonlinear constraints, multiple runs of the SUP-INF method may be required for detecting inconsistency. The following example, which is a modification of the above example, illustrates this.

$x \geq 12, \quad y \geq 1, \quad z \leq -7, \quad u \leq 3,$
$2zu \geq 3x, \quad yzu \leq 12, \quad (yz + zu) \leq -4$

After the first run, the bounds obtained are:

$12 \leq x \leq \infty, \quad 1 \leq y \leq \infty,$
$-\infty \leq z \leq -33, \quad -12/7 \leq u \leq 0.$

And, in the next run, the inconsistency is detected.

It is not clear whether for each problem, there is a suitable order in which variables can be processed to get the best results from Brooks' extension in a single run.

4.4 Splitting intervals or case analysis

As the above examples illustrate, in the case of nonlinear inequalities, the SUP-INF method may not generate good bounds on variables in a single run. It also turns out that there are examples for which better bounds exists (in the form of more than one interval), but the SUP-INF method cannot find it. In such cases, even multiple runs may not be helpful. For the following example,

$x \geq -1, y \geq 1, 3 \geq z, xy = 4x, yz = 2z.$

When the Gröbner basis algorithm is run first to process equality constraints, an additional equality constraint $xz = 0$ is generated. From these constraints, the SUP-INF method cannot really get any good bounds; it produces the following results:

$-1 \leq x \leq \infty, 1 \leq y \leq \infty,$ and $-\infty \leq z \leq 3$

However, better bounds can be obtained if the above interval for each variable is further split into subintervals. We have incorporated this into the combined method. So if $xz = 0$, then case analysis can be performed in a way similar to natural deduction method or semantic tableaux method for theorem proving in propositional calculus, and the problem is split into two subproblems - corresponding to $x = 0$ or $z = 0$.

For the case when $x \neq 0$, then $y = 4$ and $z = 0$. If $x = 0$, however, and if $z \neq 0$, then $y = 2$. For the case when $x = 0$ and $z = 0$, then $1 \leq y \leq \infty$.

Splitting a satisfiable interval of a variable is often helpful in refining the bounds of variables as there can be subintervals for variables over which the constraints may not be satisfiable.

Performing case analysis (or splitting) based on parity usually reduces the number of times the combined method may need to be run in order to detect the inconsistency or get good bounds.

We have so far experimented with case analysis on variables; it is possible to extend that analysis to expressions or terms.

5 Imprecise Data and Parameterized Models

The main motivation for implementing Brooks' method for reasoning about inequalities is to handle errors in image coordinates as well as to have the ability to specify partial information about models. An example of the latter case is that of a partial three dimensional model of some object where some of the vertices are not explicitly specified, but rather constrained by a set of relationships on model parameters. A typical example could be a building with unknown height, but known base.

Consider, for a simple example, a rectangular face with unknown height and width, but with bounded area. If this face lies in the plane $z = 10$, then the following equations define this parameterized model:

$x1 = -h, \qquad y1 = -h, \quad z1 = 10,$
$x2 = w, \qquad y2 = -h, \quad z2 = 10$
$x3 = -w. \qquad y3 = h, \quad z3 = 10.$
$x4 = w, \qquad y4 = h, \quad z4 = 10,$
$90 \leq hw \leq 110$

We would like to find whether a given image can possibly be an image of an instance of the parameterized model. Furthermore, if this is indeed an image of the model, we want to determine the transformation from the model coordinate system to the image coordinate system as well as the unknown parameters h and w. In the general case, we will not be able to completely determine these unknowns, but only further refine them, just as in the view consistency problem, a single assignment between vertex groups did not completely determine the vertex depths.

In the experiments that we performed the image coordinates were:

$u1 = -176707/12500,$	$v1 = -204131/25000,$
$u2 = -353301587/25000000,$	$v2 = 815939/100000,$
$u3 = 26413/25000000,$	$v3 = -61/100000,$
$u4 = -43/12500,$	$v4 = -408131/25000.$

These coordinates result when $h = 10$ and $w = 10$ in the above model and we rotate the model by 45 degrees around the x-axis, -35.26 degrees around the y-axis, and -30 degrees around the z-axis.

However, a priori we do not know the above exact coordinates, but only approximations to them. So we represented them as lying in intervals bounded by two rational numbers.

In order to test our preliminary implementation of the combined method, we tried several simple experiments on the model and image described above. These experiments are of the following form:

1. We start with the above equations describing the model as well as the equations describing the affine projection.
2. We also include a partial specification of the transformation between the model and image.
3. We run the Gröbner basis algorithm on this set of equations. The result is a set of equations relating transformation and model parameters to image coordinates.
4. We next add inequality constraints specifying the range of values for each image coordinate; we also add the inequality constraining the model parameters.
5. We run the SUP-INF algorithm on these constraints, specifying the image coordinate variables as input variables.

The goal in each case is to find whether (i) the above image is indeed an image of an instance of the parameterized model, and (ii) if so, to find tight bounds for the unknown parameters.

The specific experiments follow:

Experiment 1: The transformation is completely specified by

$$cos\psi = .707, sin\psi = .707,$$
$$cos\theta = .816, sin\theta = -.577,$$
$$cos\phi = .866, sin\phi = .5,$$

and the affine scale factor is 1. Except for the constraint on the area of the face, this gives a completely linear set of inequalities. The SUP-INF algorithm quickly determined the values of h and w as 10 within an error of ±0.5.

Experiment 2:

Except for the affine scale factor, the transformation parameters are known. The constraints in this case become predominantly nonlinear. The SUP-INF algorithm successfully determined tight bounds for h and w; in addition, the scale was found to be 1 within an error of ± 0.1

Experiment 3:

The rotation angle around the z-axis is unknown. Constraints now become quite complicated non-linear inequalities. Our initial experiment failed to refine the bounds for the unknown parameters although the constraints were more than sufficient to do so. We conjecture that this reflects a deficiency in Brook's extension to the SUP-INF method, but have not yet verified this.

In order to obtain some bounds, we represented the unit circle, $sin^2(\phi) + cos^2(\phi) = 1$ by a piece-wise linear approximation. To compensate for this approximation we changed the previous equalities describing the relation between the model and image coordinates into approximate equalities, with an error equal to $\pm\Delta$. With this approximation, the SUP INF method succeeded as long as Δ was sufficiently large.

Experiment 4:

We left one parameter, scale, in the transformation unknown, and dropped the image coordinates of one vertex to simulate a missing vertex in the image. This experiment is the the same as Experiment 2, except the values for $u4$ and $v4$ are unknown.

When the parity of $u4$ and $v4$ is provided, the SUP-INF method found the values for h and w within an error of ±2. Using these bounds, we were able to determine the bounds for $u4$ and $v4$.

References

Arnon, D.S., Collins, G.E., and McCallum, S., "Cylindrical Algebraic Decomposition I: The Basic Algorithm," *SIAM J. of Computing* 13, 865-877, 1984.

Arnon, D.S., Collins, G.E., and McCallum, S., "Cylindrical Algebraic Decomposition II: An Adjacency Algorithm for the Plane," *SIAM J. of Computing* 13, 878-889, 1984.

Barry, M., Cyrluk, D., Kapur, D., and Mundy, J., "A Multilevel Geometric Reasoning System for Vision," *Proc. of an NSF Workshop on Geometric Reasoning*, Oxford, England, June 1986. To appear in a special issue of the *Artificial Intelligence* Journal.

Besl, P., and Jain, R., "Three-Dimensional Object Recognition," *Computing Surveys* 17 (1), 1985.

Bledsoe, W.W., "A new method for proving certain Pressburger formulas," Advance Papers, *Fourth Intl. Joint Conf. on Artificial Intelligence*, Tibilisi, Russia, Sept. 1975, 15-21.

Brooks, R.A., "Symbolic Reasoning Among 3D Models and 2D Images," *Artificial Intelligence* 17, 1981.

Buchberger, B., *An algorithm for finding a basis for the residue class ring of a zero-dimensional polynomial ideal.* (in German) Ph.D. Thesis, Univ. of Innsbruck, Austria, 1965.

Buchberger, B, "A criterion for detecting unnecessary reductions in the construction of Gröbner bases," Proc. of *EUROSAM 79.* Lecture Notes in Computer Science 72, Springer Verlag, 3-21, 1979.

Buchberger, B "Gröbner bases: An algorithmic method in polynomial ideal theory," in: N.K. Bose (ed.) *Multidimensional Systems Theory*, Reidel, 184-232, 1985.

Connolly, C.I. and Stenstrom, J.R., "Construction of Polyhedral Models from Multiple Range Views," *Proc. 8th International Conference on Pattern Recognition*, 1986.

Cyrluk, D., Kapur, D., Mundy, J., and Nguyen, V., "Formation of partial 3D models from 2D projections - an application of algebraic reasoning," *1987 DARPA Image Understanding Workshop*, Feb. 1987, Los Angeles, Calif.

Herman, M., *Representation and Incremental Construction of a Three-Dimensional Scene Model*, Carnegie-Mellon University Report, CMU-CS-85-103, 1985.

Käufl , T., "Program verifier Tatzelwurm: Reasoning about systems of linear inequalities," Proc. of *Eighth Intl. Conf. on Automated Deduction*, Oxford, England, July 1986.

Kapur, D., Musser, D.R., and Narendran, P., "Only prime superpositions need be considered in the Knuth-Bendix completion procedure," Unpublished Manuscript, General Electric Corporate Research and Development, Schenectady, New York. To appear in *J. of Symbolic Computation*, 1985.

Kapur, D., Sivakumar, G., and Zhang, H., "RRL: A Rewrite Rule Laboratory," *Proc. of 8th Intl. Conf. on Automated Deduction (CADE-8)*, Oxford, England, Lecture Notes in Computer Science 230, Springer Verlag, 1986.

Kapur, D., "Geometry theorem proving using Hilbert's Nullstellensatz," Proc. *1986 Symposium on Symbolic and Algebraic Computation (SYMSAC 86)*, Waterloo, Canada, 202-208, 1986.

Knuth, D., and Bendix, P., "Simple word problems in universal algebras," in: J. Leech (ed.) *Computational Problems in Abstract Algebras*, Pergamon Press, 1970.

Sacks, E.P., "Hierarchical inequality reasoning," Proc. *the National Conf. on Artificial Intelligence*, AAAI, 1987, 649-654.

Shostak, R., "On the SUP-INF method for proving Pressburger formulas," *J. ACM* 24 (4), Oct. 1977, 529-543.

Shostak, R., "Deciding linear equalities by computing loop residues," *J. ACM* 28 (4), Oct. 1981, 769-779.

Strat, T., "Spatial Reasoning From Line Drawings of Polyhedra," Proc. of *IEEE 1984 Workshop on Computer Vision Representation and Control.*

Sugihara, K., "An Algebraic Approach to Shape From Image Problems," *Artificial Intelligence* 23, 1984.

Thompson, D. and Mundy, J., "Three-Dimensional Model Matching From an Unconstrained Viewpoint," Proc. of *IEEE Conf. on Robotics and Automation*, April 1987.

van der Waerden, B.L., *Modern Algebra, Vol. I and II.* Frederick Ungar Publishing Co., New York, 1966.

Test Results from SRI's Stereo System

Marsha Jo Hannah
Artificial Intelligence Center, SRI International
333 Ravenswood Avenue, Menlo Park, CA 94025

Abstract

This paper briefly describes the automatic system, STEREOSYS, which has been developed at SRI for stereo compilation. This system uses area-based correlation, but applies this basic technique in a variety of novel ways to develop a disparity model for a given stereo image pair. The techniques used are hierarchical in nature, and incorporate iterative refinement, as well as a best-first strategy, in the matching process. This paper also presents results obtained with this system on test data recently distributed by the International Society of Photogrammetry and Remote Sensing, Working Group III/4 (Mathematical Aspects of Pattern Recognition and Image Analysis) as part of their Test A on Image Matching. ISPRS has not yet completed detailed analysis of these results, but preliminary indications are that STEREOSYS performed better than all other tested algorithms.

Introduction

Over the past several years, SRI has integrated and improved existing pieces of stereo software to form STEREOSYS, a baseline system for automated, area-based stereo compilation [Hannah, 1985]. This system operates in several passes over the data, during which it uses several different matching algorithms iteratively to build and refine its model of a portion of the three-dimensional (3-D) world represented by a pair of images. The underlying strategy is to begin with a few points that are most likely to be matchable (based on their "interest", i.e. information content); these are matched by very global, but very conservative, search algorithms. Each successive algorithm operates on less promising points, but uses more information from matches made at previous levels to constrain the search to smaller and smaller portions of the epipolar line, until eventually all interesting points have been processed.

In November of 1986, we were invited to join an international group of photogrammetrists and computer vision researchers participating in a test of image matching capabilities. Each participant received the same 12 pairs of digital images, which he was asked to process automatically, returning the results to the International Society of Photogrammetry and Remote Sensing's Working Group III/4 (Mathematical Aspects of Pattern Recognition and Image Analysis) for evaluation. Participants had the options of producing matches on a specified regularly spaced grid of points in the left image (Task A), of selecting left-image points and matching them (Task B), or of performing other specified measurement tasks that varied from one image pair to another. This paper briefly describes STEREOSYS and presents its test results.

Description of STEREOSYS

The matching algorithms in STEREOSYS begin by selecting a set of well-scattered windows in one image, such that each window contains sufficient information to produce a reliable match. To accomplish this, a statistical operator is passed over the image; this operator is a product of the image variance and the minimum of ratios of directed differences (hence edge strength) over windows of the specified size [Hannah, 1980]. Local peaks in the output of this operator (the "interesting" points) are recorded as the preferred places to attempt the matching process. To ensure that the selected windows are well-scattered in the image, the image is divided into a grid of subimages, and the relative ranks of the interesting points within their grid cell are recorded; this permits the most interesting points in each area to be matched first.

Whether or not point (x_1, y_1) in the first image I_1 is matched by point (x_2, y_2) in the second image I_2 is determined by computing the cross-correlation, normalized by both mean and variance, over windows surrounding the points [Hannah, 1974]. The matching point is taken to be the point in I_2 with highest correlation, as located by one of several search algorithms. All of our matching algorithms use image hierarchies to some extent. Pixels in each reduced image of the hierarchy are produced by convolving the parent image with a Gaussian, then sub-sampling [Burt, 1980].

The first matching algorithm, unconstrained hierarchical matching, matches each specified point (usually the most interesting point in each grid cell) using an unguided hierarchical matching technique [Moravec, 1980]. This technique permits the use of the overall image structure to set the context for a match; the gradually increasing detail in the imagery is then followed down through the hierarchy to the final match.

In this matching technique, as in all the others we use, matches must pass fairly strict tests in order to be considered correct. At any level in the hierarchy, matches with poor correlation are discarded, as are matches that fall outside of the image. Each match must also be confirmed by back-matching, that is, if we have found that point (x_1, y_1) in the first image I_1 is best matched by (x_2, y_2) in the second image I_2, we then repeat the entire matching algorithm, starting with (x_2, y_2) in I_2 and searching for the point (x_1', y_1') in I_1 that best matches (x_2, y_2). If (x_1, y_1) and (x_1', y_1') differ by more than one pixel, the entire match is discarded as being unreliable.

If no *a priori* absolute camera model has been given with the data set, we next calculate a simplistic relative camera model from the set of point pairs produced by unconstrained hierarchical matching. This is accomplished by searching for five angles that best describe the relative positions and orientations of two ideal pinhole cameras [Hannah, 1974].

The next technique to be applied is epipolar constrained hierarchical matching. Having determined the camera parameters, we now know the manner in which a point in the first image projects to a line in the second image—the epipolar constraint. This constraint allows us to cut the search from two dimensions (all around the point) to one dimension (back and forth along the epipolar line) at each level of the hierarchy. In all other respects, epipolar constrained hierarchical matching proceeds very much like unconstrained hierarchical matching, and is used on any unmatched points amongst the two most interesting points for each grid cell.

Once a good basis of reliable matches has been found, these matches can be used as "anchor" points for the anchored matching technique. Under the assumption that the world is generally continuous, a point would be expected to have a depth similar to that of its neighbors. Thus, the disparity for a point is expected to lie in the interval of the disparities of the well-matched points in the current and neighboring cells in the grid of subimages. This disparity interval is used along with the epipolar constraint to perform a very local search for the match to a point, perhaps proceeding one or two levels up the image hierarchy, to provide context for the match.

Our system also can produce matched points on a regularly spaced grid, if desired. This matching algorithm also uses the anchored match technique, searching along specified portions of the epipolar line, to calculate matches for the user-specified grid of points in the first image. However, holes can result if a grid point does not have suitable information for matching, and only matches that pass all the tests are recorded.

Results on Image Matching Test A

For each of the 12 image pairs in ISPRS's Test A on Image Matching, we attempted to perform Standard Task B—determination of the parallaxes at selected points—which is what STEREOSYS does best; for a few of the images, we also attempted to determine the parallaxes at a grid of points. In most cases, these 240x240 images were run with the standard parameters for the system, which had been tuned to process a high-quality, 1024x1024 aerial image pair. Because of incompatibilities in format, we did not use the camera orientation information given with each test image pair, or any other *a priori* information; we used the raw images, without transforming to normal images, or doing any other resampling.

For the most part, the matching proceeded routinely, using the standard procedures and parameters. One parameter—a threshold that indirectly controls the number of interesting points that the system has to work with—was changed for each data set; this was done to produce between 100 and 300 points per image, as requested for the test. In what follows, we outline processing steps that deviated from the usual and comment on the overall results. In the figures that illustrate our results, matches are shown with one of two different marks; the smaller marks correspond to matches with correlations less than 0.3 . The system has discarded matches that it was not confident about, but we have done no other editing of its final results.

On the images for Test 1 (named Car I—part of the engine compartment of a car, sprayed with undercoating), the system produced 335 interesting points, of which it matched 330 (Figure 1). We also attempted to match a grid of points, which produced 507 matches (Figure 2). The interesting points are fairly well scattered throughout the image, providing a good base of matches for further processing. The grid of points is fairly complete. From a brief visual inspection, all of the matches appear to be reasonably correct.

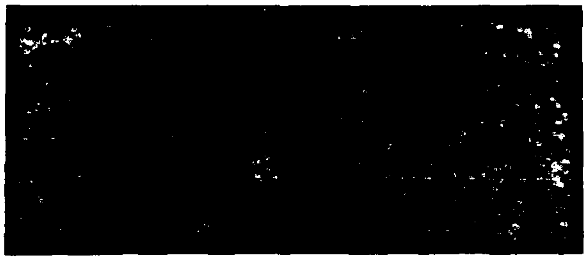


Figure 1: Test 1 (Car I)–Parallaxes at selected points.

Figure 2: Test 1 (Car I)–Parallaxes at a grid of points.

On Test 2 (Quarry—an aerial veiw of a rock quarry), the system produced 275 interesting points, of which it matched 249 (Figure 3). We also attempted to match a grid of points, which produced 437 matches (Figure 4). Again, the interesting points are fairly well scattered throughout the image, providing a good base of matches for further processing, although there are a few blank spots. The grid of matches has more holes than the previous example, but is still fairly complete. All of the matches appear to be reasonably correct. The foreground post was found to be interesting, but those matches were at the disparity of the background, with degraded correlations.

Figure 3: Test 2 (Quarry)–Parallaxes at selected points.

On Test 3 (Olympia I—an oblique view of part of the plexiglas roof of the Olympia tent in Munich), the system produced 171 interesting points, of which it matched 139 (Figure 5). Because of the sparse information in this image, we did not attempt to produce matches on a grid. Our system had considerable trouble with this image pair, because of the general lack of information other than the ambiguities caused by the many identical buttons. Performing unconstrained hierarchical matching on just the single most interesting point in each grid cell did not produce enough matches to form a camera model, so we started over, asking for

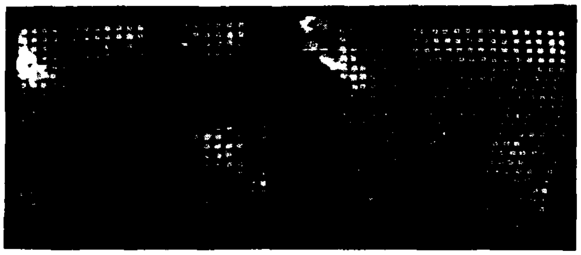


Figure 4: Test 2 (Quarry)-Parallaxes at a grid of points.

the four most interesting points per cell to be attempted. Of the 25 matches produced this way, one was clearly wrong; the camera model solver was unable to converge to a model, and hence was unable to edit out the point. In this instance (the only time we did so for any test), we removed the offending point by hand, then tried the camera model solution again, with good results. (If our model solver included the RANSAC technique [Fischler and Bolles, 1980], we believe it could have proceeded without intervention.) To be consistent with the unconstrained matching, the epipolar hierarchical matching was also done for any as-yet unmatched points amongst the four most interesting points per cell, rather than the usual two most interesting points. Doing an anchored match with the normal parameters seemed to miss a lot of points, so we asked for a second pass of anchored matching, using all previous matches as anchors, doubling the number of grid cells in each dimension, and constraining the search to just the highest level of the image hierarchy. This produced a fairly good base of matches, which appear to be reasonably correct.

Figure 5: Test 3 (Olympia I)-Parallaxes at selected points.

On Test 4 (South America—an aerial photo of an area without vegetation), the system produced 309 interesting points, of which it matched 169 (Figure 6). We also attempted to match a grid of points, but this resulted in more "holes" than "grid", so we aborted the effort. The matched points are fairly well scattered throughout the image. Because of the poor image quality, most of the correlations were fairly low, which accounted for most of the points that the system said could not be matched. The better matches that were retained appear to be mostly correct, although some of the matches appear to be off by a little. Because of the fuzziness of the images, it is difficult to assess the correctness of the matches.

On Test 5 (Bridge—part of a suspension bridge in the Grand Canyon), the system produced 219 interesting points, of which it matched 180 (Figure 7). Based on our difficulties with Test 3, we began by trying to match the 2 most interesting points in each cell instead of just the single most interesting point per cell, as would be usual. That worked quite well (i.e. the system probably would have worked correctly with its normal settings), and the system operated routinely from there. The matched points are fairly well scattered along the "wall" of the bridge, and are strung along the lines on the "floor" of the bridge. The matches appear to be reasonably correct, although a couple of the matches appear to be off by one wire in the fence, or by one grid element in the floor, because of the ambiguities of repetitive patterns.

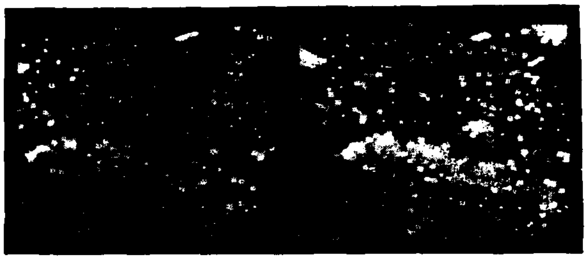


Figure 6: Test 4 (South America)-Parallaxes at selected points.

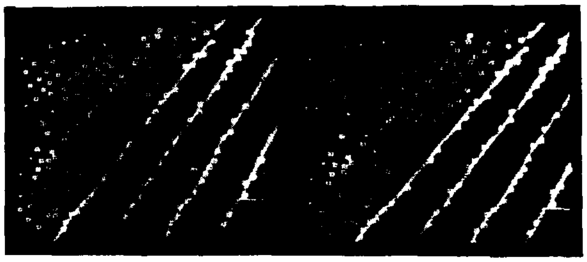


Figure 7: Test 5 (Bridge)-Parallaxes at selected points.

On Test 6 (Tree—an assortment of tree branches against a background of sky), the system produced 158 interesting points, of which it matched 109 (Figure 8). The interesting points are strung out along the various branches, marking some but not all of the branch forks and crooks. A few "interesting points" are off to the side of a branch, rather than centered on it; this is a known weakness of our interest operator. Despite the unusual nature of the scene, it was processed with the usual parameters and sequencing of routines. Most of the matches are locally plausible, although some of them have tracked "false intersections"—places where one branch passes in front of another, rather than an actual real-world point.

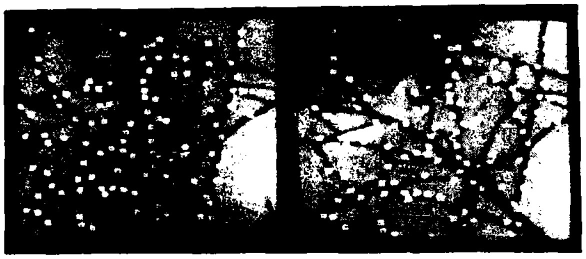


Figure 8: Test 6 (Tree)-Parallaxes at selected points.

On Test 7 (Island—an aerial view of a rocky island off the coast of Sweden), the system produced 311 interesting points, of which it matched 260 (Figure 9). The interesting points are fairly well scattered throughout the image, providing a good base of matches for further processing. Most of the matches appear to be reasonably correct, although there are a few questionable ones near the edges of the image.

On Test 8 (Switzerland—an aerial view of a smooth hill with scattered trees, taken under different lighting conditions), the system produced 279 interesting points, of which it matched

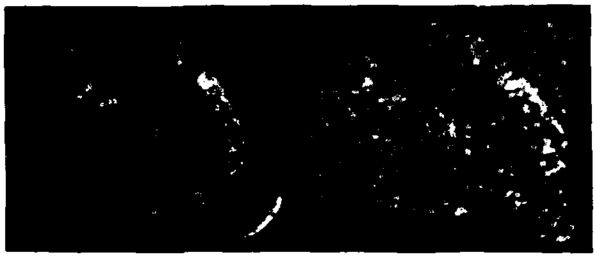


Figure 9: Test 7 (Island)–Parallaxes at selected points.

196 (Figure 10). The interesting points are fairly well scattered throughout the image. The matches appear to be substantially correct, although they represent a mixture of tree tops and ground points, since our system is unable to distinguish between the two. Many of the correlations are low, possibly because the two images appear to have been recorded at significantly different times of day.

Figure 10: Test 8 (Switzerland)–Parallaxes at selected points.

On Test 9 (Car II—another view of the car's engine compartment), the system produced 271 interesting points, of which it matched 168, using the usual processing routine and parameters, despite the wide range of disparities (Figure 11). The matched points are concentrated in a few clusters on high-intensity areas of the image. It is difficult to fuse these images long enough check the results, but most of the matches appear to be approximately correct, with the possible exception of a few matches near the headlight.

Figure 11: Test 9 (Car II)–Parallaxes at selected points.

On Test 10 (Wall—an oblique view of the working face in a stone quarry), the system produced 224 interesting points, of which it matched 194 (Figure 12). The matched points are fairly well scattered throughout the image, avoiding blank spots and the very linear edges between faces. The matches appear to be reasonably correct, with a few possible problems near the edge of the image.

On Test 11 (Olympia II—another view of the Olympia tent), the system produced 197 interesting points (Figure 13). After the difficulty we had with Test 3 (Olympia I), we tried matching the four most interesting points in each cell. Of these 133 points, only 4 resulted in matches that our system thought were reliable, based on its local criteria, and none of these matches were actually correct. We therefore abandoned processing on this image pair. We believe that the combination of repetitive structures, the rather different points of view, and the transparency of the dome caused our hierarchical matching techniques to fail. If we had elected to use the accompanying camera information, we might have been able to do some matching, although the transparency would undoubtedly have lead to numerous problems.

Figure 12: Test 10 (Wall)–Parallaxes at selected points.

Figure 13: Test 11 (Olympia II)–Unable to determine parallaxes.

On Test 12 (House—an aerial view of a house and fence on rolling terrain), the system produced 119 interesting points, of which it matched 92 (Figure 14). The interesting points are poorly distributed throughout the image, avoiding the featureless areas of the driveway, lawn, and roof, and the linear edges of the shadow. The matches appear to be reasonably correct, even in the difficult area where the back of the house falls off to the ground.

Figure 14: Test 12 (House)–Parallaxes at selected points.

Summary

In this paper, we have described STEREOSYS, our automatic system for stereo compilation, which uses area-based correlation, but applies this basic technique in a variety of novel ways. Our techniques are hierarchical in nature, and use iterative refinement with a best-first strategy, in the matching process, as well as the constraint of backmatching to verify matches. We have also presented the results of our system on the Image Matching Test A data set recently distributed by ISPRS's Working Group III/4. Our results on ISPRS's test imagery have again demonstrated the robustness of the correlation-based matching technique, and its wide range of applicability over image types.

Overall, we are very pleased with the results of our system. For 10 of the 12 tests, we were able to handle the image pairs without substantially altering the default parameters or processing sequence. Our only problems came on the images of the Olympia dome; this is not surprising, since our algorithms were designed for use on highly textured natural terrain, not the bland faces and transparency of cultural objects. For the most part, our results appear to be reasonably correct, even in the face of large disparity ranges within small areas of the image. As of this writing, we have not received the results of the committee's detailed analysis, however, we have been told that our algorithm was able to process a larger number of the images than any other participant.

Acknowledgements

The research reported herein was supported by the Defense Advanced Research Projects Agency under Contracts MDA903-83-C-0027 and MDA903-86-C-0084, which were monitored by the U.S. Army Engineer Topographic Laboratory. The views and conclusions contained in this paper are those of the author and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or of the United States Government. I would like to thank Harlyn Baker, Lynn Quam, Grahame Smith, and Martin Fischler for their support on this project.

References

Burt, P. J., 1980. "Fast, Hierarchical Correlations with Gaussian-like Kernels," University of Maryland Computer Science Center Report TR-860, January, 1980.

Fischler, M. A. and Bolles, R. C., 1980. "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography," *Proceedings: Image Understanding Workshop,* College Park, MD, April, 1980, pp. 71-88.

Hannah, M. J., 1974. "Computer Matching of Areas in Stereo Images," Ph.D. Thesis, Stanford University, Computer Science Department Report STAN-CS-74-438, July, 1974.

Hannah, M. J., 1980. "Bootstrap Stereo," *Proceedings: Image Understanding Workshop,* College Park, MD, April, 1980, pp. 201-208.

Hannah, M. J., 1985. "SRI's Baseline Stereo System," *Proceedings: Image Understanding Workshop,* Miami, FL., December, 1985, pp. 149-155.

Moravec, H. P., 1980. "Obstacle Avoidance and Navigation in the Real World by a Seeing Robot Rover," Ph.D. Thesis, Stanford University, Computer Science Department Report STAN-CS-80-813, September, 1980.

PRELIMINARY DESIGN OF A PROGRAMMED PICTURE LOGIC

David Harwood
Raju Prasannappa
Larry Davis

Computer Vision Laboratory
Center for Automation Research
University of Maryland
College Park, MD 20742-3411

ABSTRACT

The objective of the PPL project is to design and implement a general and modular logic-programmed system for two-dimensional interpretation of image theories in image structures obtained by image analysis. Important subsystems include heuristic search for object instances with optimization of goodness-of-figure, and procedures for computing basic image components, locales for searches, and predicates. We illustrate some of these in an application to aerial images of suburban neighborhoods.

1. INTRODUCTION

Image analysis and interpretation are the principal objectives of computer vision. This report describes a modular logic-programming system, called Programmed Picture Logic (PPL), for two-dimensional interpretation of images, and illustrates its application to aerial photos of suburban neighborhoods.

The goal of PPL is to specify and implement a "*constructive model theory*" *for logical theories* of images. The approach is general—essentially logic-programmed "generate and test"—which combines symbolic and numerical programs of Prolog, Lisp, and C for logical rep.esentation and inference, hierarchical segmentation and search, procedural evaluation of image predicates, and optimization of figures. The emphasis in this research has been on careful general, even simple, design rather than upon specialized techniques; this may be more effective with the advances in supercomputing.

Important related research in the same area of expert interpretation of aerial photos is by McKeown (1985) who has developed a rule-based system for image interpretation, using regional segmentation and search, which is less complete than that of PPL. Selfridge (1982) uses adaptive thresholding for image resegmentation, while PPL uses type-specific, multi-level connected component analysis. Rule-based systems for uninterpreted image segmentation have been developed by Nazif and Levine (1984); however, there are very few large applications of logic programming to expert computer vision. Recently, Reiter and Mackworth (1987) have discussed application of formal logic in a theory of depiction and interpretation, although without implementation by programs. [Maier and Warren (1988) is an excellent introduction to logic programming, generally, besides Prolog; Bratko (1986) focuses on AI algorithms; van Caneghan and Warren (1986) gives other applications of logic programming.]

2. OVERVIEW OF PPL

The current version of PPL is a mixed-language system which uses Prolog for transparent definition of semantic types and relations of objects and for control of image analysis and interpretation, and which efficiently evaluates image predicates by calls to Lisp or C functions.

The goal of the PPL system is to construct a largest consistent interpretation or model of the Prolog "image theory", subject to constraints on computational complexity, by searching for instantiations within certain combinatorial topologies obtained by four kinds of combinatorial operations on connected components. The search attempts both to find instances and to delineate them accurately. The complexity of search is reduced by a number of means which will be discussed later.

The "philosophy" of PPL is essentially one of sophisticated, logic-programmed generate-and-test. The block diagram of the system, shown in Figure 1, will be useful in our discussion of what this means.

2.1. Preprocessing and Segmentation

The input image is first enhanced before segmentation. This involves recursive application of a Symmetric Neighborhood Filter (SNF) and optimization of the resulting near fixed-point image with respect to the original image until a more accurate enhanced image is obtained (Figure 2). Currently, this same enhanced image is used throughout resegmentation, so the result of interpretation depends very much on the filter's quality. We present results of enhancing poor aerial images of suburban neighborhoods. Also, Figures 3a-b show the results of enhancing a complete aerial photograph with very fine structure which is obliterated by previously developed smoothing operations.

THE PROGRAMMED PICTURE LOGIC (PPL) SYSTEM

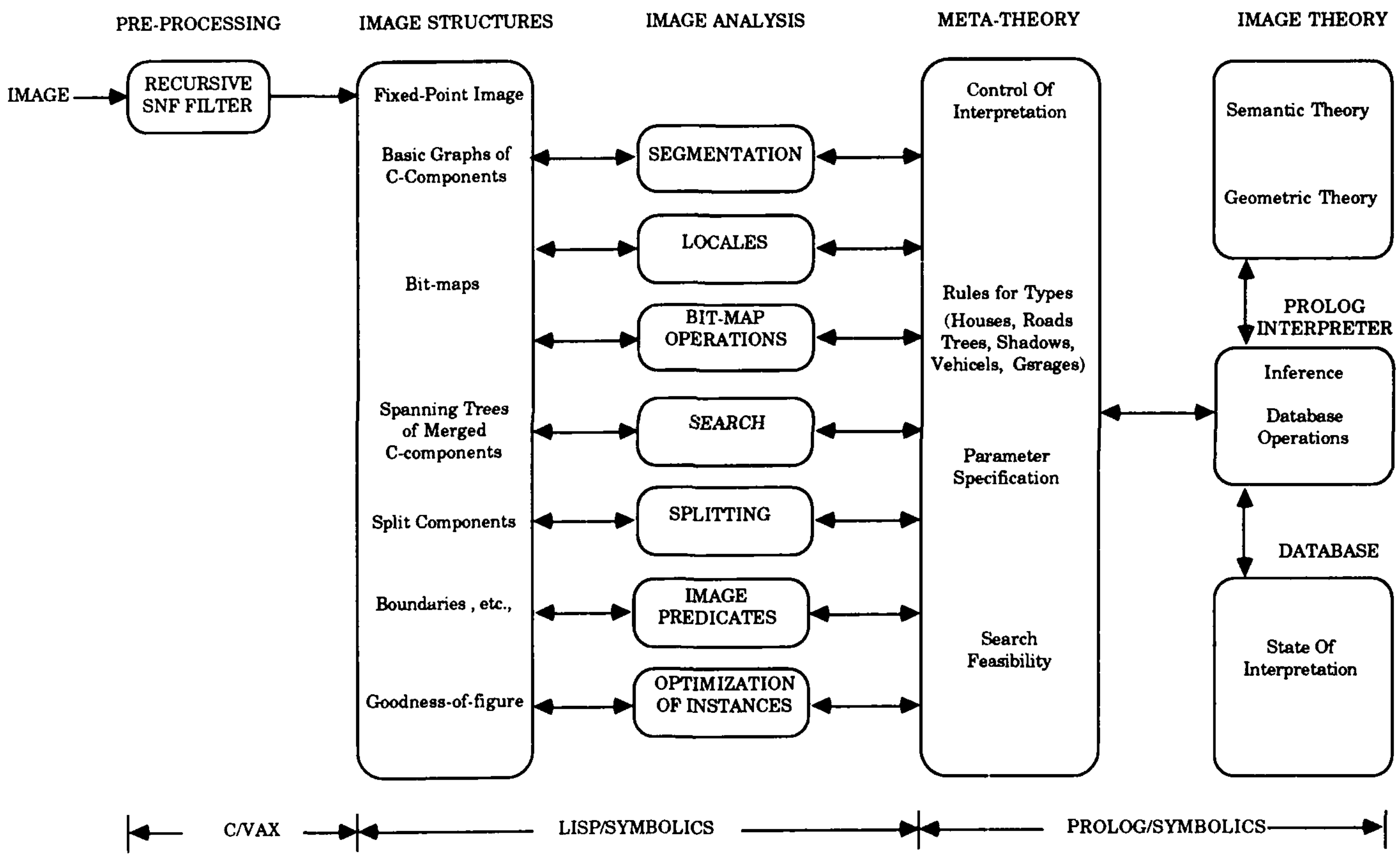


Figure 1. PPL system diagram.

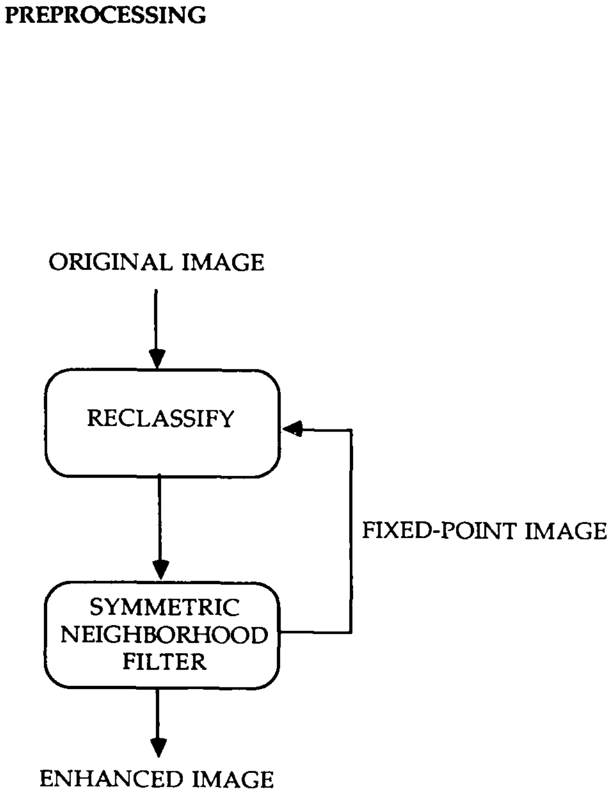


Figure 2. Preprocessing.

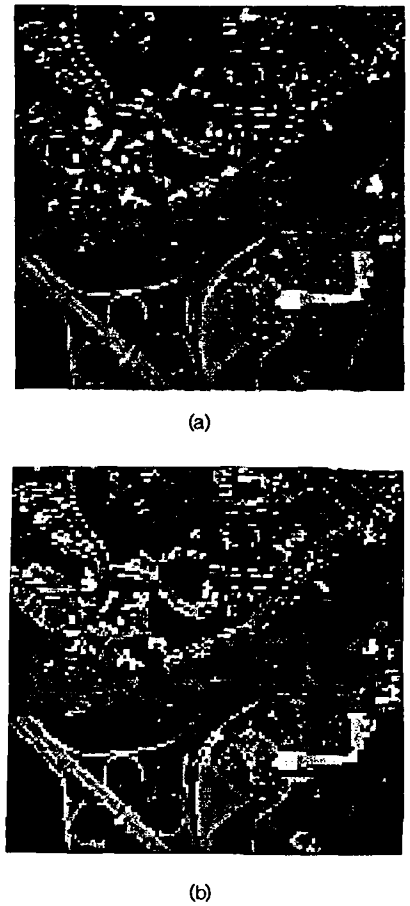


Figure 3. (a) Original image. (b) SNF, fixed point.

The enhanced image is segmented by a simple gray-level connected component algorithm into a hierarchical set of segmentations consisting of increasingly finer connected components (Figure 4). The finest level of segmentation is called the atomic segmentation and the components at that level are called the atomic components. The components at coarser levels of segmentation are represented as compositions of these. [We admit the possibility that atomic regions may sometimes have to be split by geometrical region-splitting operations. We can also regard the connected component algorithm as a kind of splitting of larger components according to gray-level contrast.] We maintain a unique symbolic representation of every component at every level of segmentation, which also allows efficient symbolic implementations of some of the combinatorial and set operations on regions, e.g., intersection. In the current implementation the segmentations are generated before the interpretation starts.

Next, labeled adjacency graphs of segmentations are constructed for use during search. Some of the properties of the connected components frequently used by the search process are also computed: mean gray level, area, and contrast at boundaries. These graphs will also be used in an analysis of whether a goal is satisfiable at all, and to determine whether a complete search is feasible. If it is not, then a more constrained, heuristic search has to be applied. Bit-maps of components are computed only at the atomic segmentation level; the bit-maps at coarser levels are computed only when necessary. Segmentation and attribution of components is implemented in Lisp on a Symbolics computer, thus allowing an efficient implementation of heuristic graph search.

2.2. Image Theory

The image theory can be considered as the main knowledge base of the system, and consists of a set of logical formulae representing a Prolog program. It can be divided into two parts: a semantic theory and an auxiliary geometric theory. The semantic theory specifies the semantic types and relations between the important objects of the image domain. For example, in the domain of suburban neighborhoods the variety of definitions varies from simple ones such as generic

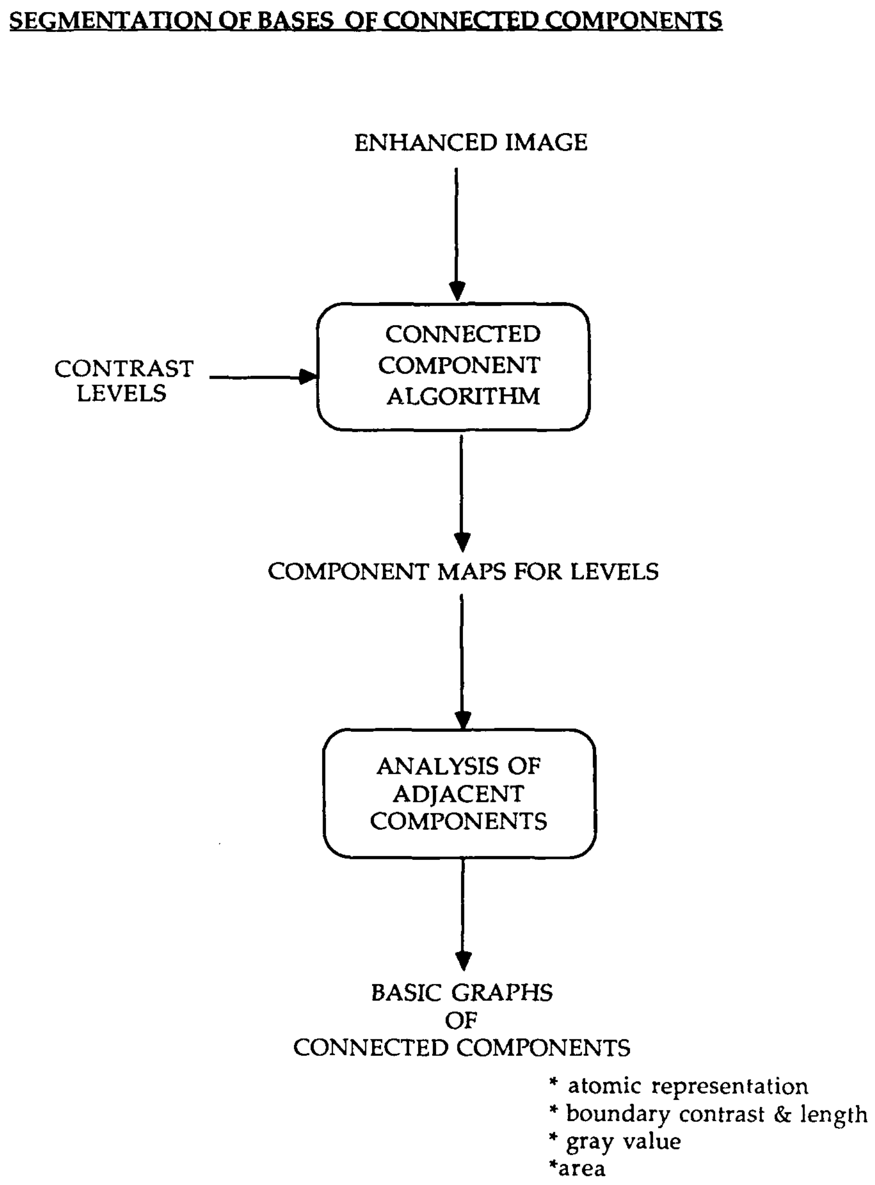

SEGMENTATION OF BASES OF CONNECTED COMPONENTS
ENHANCED IMAGE
CONTRAST LEVELS
CONNECTED COMPONENT ALGORITHM
COMPONENT MAPS FOR LEVELS
ANALYSIS OF ADJACENT COMPONENTS
BASIC GRAPHS OF CONNECTED COMPONENTS
* atomic representation
* boundary contrast & length
* gray value
*area

Figure 4. Segmentation for search.

houses, roads and trees to complex ones such as houses of different shapes (L-shaped houses, houses with chimneys, houses with patios, houses with different roofing), vehicles, or trees in the front yard of a house. [The auxiliary geometric theory consists of rules about mathematical operations on arbitrary regions or boundaries in the image; these might include, for example, simple rules for set, connectivity, angular or metrical relations.] The following types of objects are defined in the image theory of the current version of PPL: houses, house-adjuncts, shadows of houses and adjuncts, road complexes, trees, and vehicles on roads or driveways. The use of the logical representation of Prolog makes the expansion of the image theory quite straightforward.

2.3. Meta-Theory

Meta-theory is the theory about finding a logically consistent interpretation of the image theory within various analyzed image structures; it is implemented as a meta-interpreter of the image theory in Prolog. We may think of this as a program for a constructive model theory—including rules which control the construction by the Prolog interpreter of a model (or set of solutions) of a given set of axioms for images of objects. It specifies the flow of control between the different modules of the system in terms of Prolog rules for different types of objects and also provides a simple user interface for specification of top-level system goals.

queries and certain parameters associated with appearances of objects (e.g., 3D sun direction, scale, gray values). It makes use of the inference mechanism of Prolog to build a largest consistent interpretation of the image theory with respect to the image (i.e., each image of application).

Meta-theory, in addition, specifies relevant parameters and rules which constrain search for specific types of objects—what are their locations, simple properties, and expected levels of segmentation. In the current version of PPL these are predetermined by prior experiments and encoded into the image theory as Prolog facts. [It should be noted, however, that we plan to investigate interactive parameter specification by sampling instances, using the same procedures of PPL in inverse mode. With this, we may identify some instances and have the system find others iike them.]

To illustrate the actions of meta-theory (and hence the procedure of interpretation) let us consider a simple goal:

Prove that there exists an X such that X is a house.

This query can be represented in Prolog as:

house(X).

The meta-theory first tries to satisfy this goal by instantiating the variable X to the first instance of house that can be deduced from the current state of interpretation. The current state of interpretation can be obtained from the Prolog database which contains facts about that part of the image which has been interpreted so far. (Actually, the interpretation consists of these facts, plus all those which can be deduced from these using the image and geometric theories.) There are two conditions under which this goal may fail to be satisfied: the store of facts is incomplete, so no instance may be deduced, or there simply aren't any instances satisfying the definition of a house. In the latter case the system returns that the goal could not be satisfied. In the former case, which is more interesting, the meta-theory initiates a search for all possible instances. The next sub-section describes this search process in detail.

2.3.1. Search

The search module searches through the hierarchy of segmentations under a set of relaxed constraints on the properties of the object to find regions which are most likely to be instances of the object using a combination of merging, splitting, union and intersection. [There are restrictions on the latter three operations, which are necessary to avoid intractable complexity.] A block diagram of this search is shown in Figure 5. The parameters given to the search module are specific to the sought-for object type. They consist of the bounds on the segmentation levels to be searched, a set of region properties and a locale for search. The region properties determine the merging criteria and the terminating conditions for search and are determined by the meta-theory as mentioned in the last section. The properties used in the current version are expected gray level, expected area, tolerance on area and gray level, and threshold on the contrast between adjacent components. The search is conducted level by level beginning from the coarsest level of segmentation. The search space is reduced by restricting the search to a subgraph of the basic connected component graph based on the search locale. First, those regions with average gray levels between the upper and lower bounds of the expected gray level are selected to be the roots for the search process. Starting from a root, larger and larger regions are generated by merging adjacent components (sorted in increasing order of contrast along their boundaries) that satisfy the merging criteria, and at the same time checking for search terminating conditions. The merging criteria used in the current implementation are:

(1) The average contrast along the boundary between the two regions is less than the contrast threshold.

(2) The expected gray level is within tolerance values.

(3) The total area of the resulting region is within tolerance values.

This cycle of merging and testing continues until a region whose area is at least equal to the expected area and does not exceed the upper bound on area is obtained (successful search) or there are no more regions that can be merged (unsuccessful search). A search tree is generated starting from each root. If the search rooted at each of these roots fails the search is continued at the next finer level, with a new set of roots at that level. At the end of the search the search module returns a list of leaves of the successful search trees. The search module is efficiently implemented in Lisp on the Symbolics; the data structure used is a spanning tree, used to avoid redundant evaluations.

Due to the exhaustiveness of the search only those properties of regions which are relatively inexpensive to compute are used in the merging and terminating criteria. Hence the regions returned by the search module may not satisfy the definition of the object completely, but rather only a relaxed definition of the object based on gray level, area, and contrast, besides the important constraint on location with respect to other objects. Each of these regions is admissible for evaluation with respect to the full definition of the object, involving possibly any sort of image analysis. For example, compactness and rectilinearity are checked for house instances. Several Lisp functions are provided in the library of image predicates for computing some of the properties of regions used during this evaluation. The image analyses implemented in the current version are boundary extraction, compactness and rectilinearity. However, many more will have to be added as the objects in the image domain become more and more complex. All the admissible instances and any of their properties computed so far are recorded in the Prolog database as facts, so that they can be used by the meta-theory without having to recompute them if needed during further inferences.

SEARCH BY MERGING

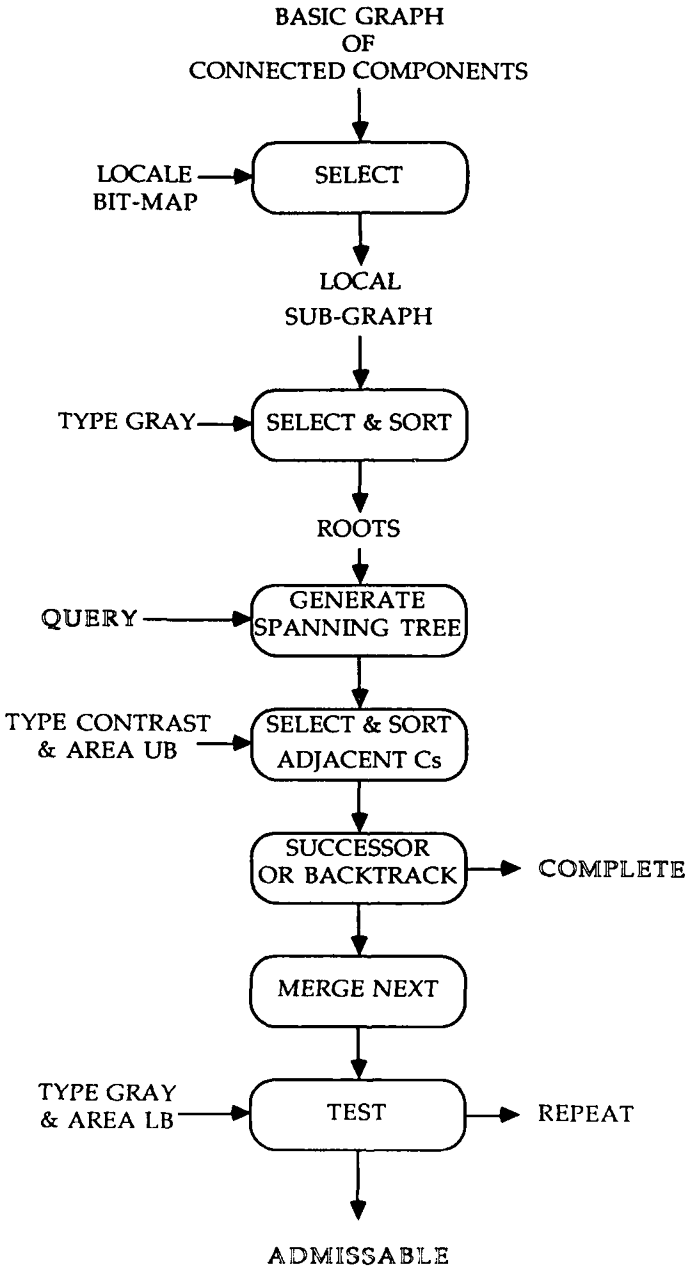


Figure 5. Search by merging.

2.3.2. Optimization

The complete search described above has the advantage that it generates every instance that could possibly be generated. However, it may also generate many spatially overlapping candidate instances of the same type of object. It is the function of the optimization process to resolve this conflict by selecting the one with the best goodness-of-figure among these overlapping instances. To check if a candidate instance Y is optimal with respect to a particular type of object it is compared with every other candidate instance of the object in the database (note that all instances are recorded in the Prolog database). If a candidate instance X overlaps with Y then their goodness-of-figure values are compared. Intersection of sets of atomic components forming a region (atomic representation) is used to check for overlapping regions (bit-maps of the regions can also be used). We have used average absolute value of figure-ground-contrast as a measure of the goodness-of-figure (besides usual definitional constraints on shape, for example, simple connectedness—no holes, or sufficiently straight edges). If the goodness-of-figure of X is better than that of Y then all facts about Y are deleted from the database and X replaces Y as the current best instance. Otherwise, all facts about X are deleted from the database. This is continued until all candidate instances of the object type have been examined, when the last is determined to be an instance having best figure. Figure-ground contrast has been found to be a reasonable measure of goodness-of-figure for most types of objects in our image domain.

3. IMPLEMENTATION AND EXPERIMENTAL RESULTS

3.1. Implementation

The PPL system is a multi-language system (C, Lisp and Prolog). Except for preprocessing, which is implemented on a VAX in C, the rest of the system is implemented on a Symbolics in Lisp. Equivalent representational primitives have been developed using Flavors. An image is represented as an instance of a flavor called PPL-image. The software required for I/O is implemented in Lisp. The segmentation and search modules are implemented in Lisp using Flavors. A basic flavor called Component is defined to represent nodes in the search tree and the graphs. Another flavor called Node is defined on top of this to represent instances of objects. The rest of the system, including the meta-theory, is implemented in Prolog. Two dialects of Prolog are available in the software environment of the Symbolics: DEC-10 Prolog and Lisp Prolog. The former was chosen for portability reasons.

As is evident from the discussion earlier, we use Prolog's backward-chaining capability for handling the flow of control between the different modules of the system. To illustrate this, let us consider a typical example. A Prolog rule for detecting shadows of simple houses might be:

```
house_shadow(HS) ← house(H),
                   house_shadow_locale(H,L),
                   candidate_in_locale(L,HS),
                   defn_shadow(HS),
                   optimal_figure(HS).
```

This rule states that HS is a shadow if there exists a house H whose shadow locale is L and HS is a good shadow lying within locale L and is an optimal instance of a shadow. The sub-goal defn_shadow(HS) will itself be another rule which defines a house shadow based on its primary features. When the meta-interpreter is asked to satisfy the goal house_shadow(HS), Prolog has to satisfy each of the sub-goals in the body of the rule in order to satisfy this goal. The Prolog interpreter tries each of the sub-goals from left to right and returns the first instance satisfying the goal. Additional instances can be obtained by asking the system to resatisfy the goal. Note that if there is no house in the database the system will try to deduce an instance of a house using the house rule. In order to extend the system to include other objects, more definitions can be added to the Prolog image theory easily without worrying too much about the underlying control structure, although we may have to add programs for new image predicates.

3.2. Object Definitions

The following are definitions of the objects which are searched for by the present implementation of PPL.

A house is a connected region of the image satisfying constraints on area, compactness, average gray value, boundary contrast, and rectilinearity. (All instances of all types must have best figure-ground contrast.) A house is rectilinear if a large proportion of its boundary tangents are parallel or normal to one direction; we also check the ratio of frequencies of tangents in the major (parallel) and minor (normal) directions.

A shadow-of-a-house is a region within a house-shadow locale satisfying constraints on gray value, contrast, and area (depending on sun inclination.)

A house-shadow locale is a subset of the image determined by the 3D sun direction, the border of a house instance, and a hypothesis of maximal house height. (We only estimate this locale roughly since we don't know the sun direction accurately.)

A house-adjunct has a definition like that of a house. Shadow-of-a-house-adjunct and its locale are similar to the corresponding concepts for houses.

A road complex has suitable gray level, area, and non-compactness. (These weak definitions have been found to work well enough for our current purposes.)

A tree has constraints on area and gray-level. (Woods are another matter, requiring a more complicated definition.)

A vehicle is a small, compact, bright region, located on or near a driveway or road. A vehicle-on-a-road locale is a subset near a road. A vehicle-on-a-driveway locale is near a house and a driveway.

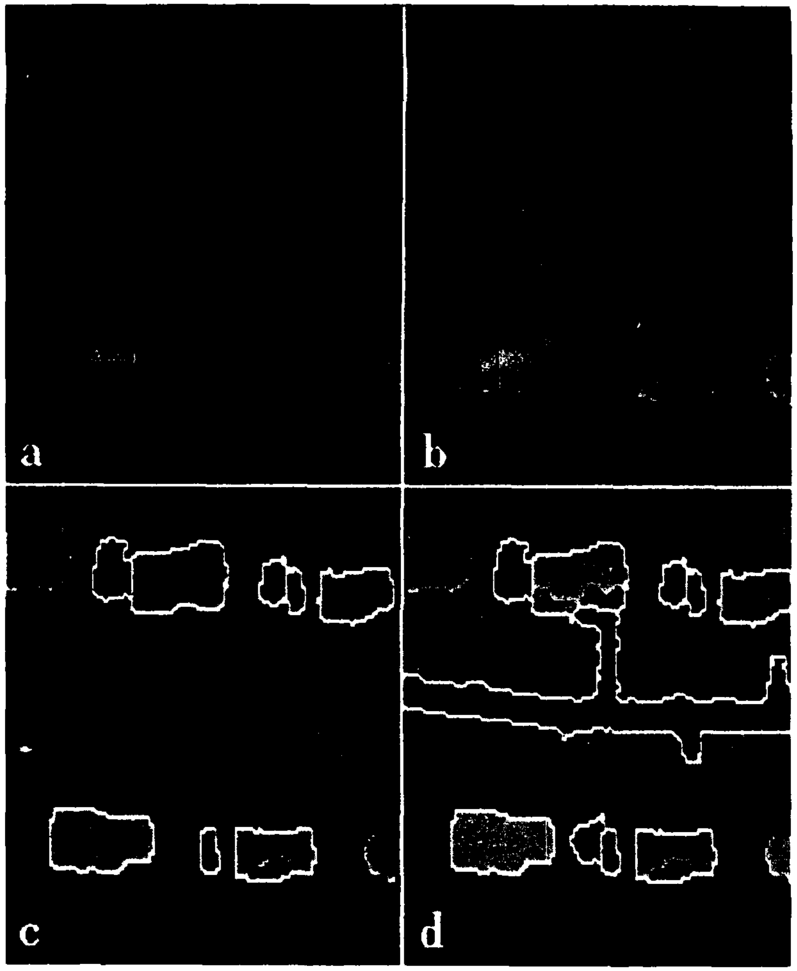


Figure 6. (a) Original image. (b) SNF eiianced image. (c) High-contrast search. (d) Low-contrast search.

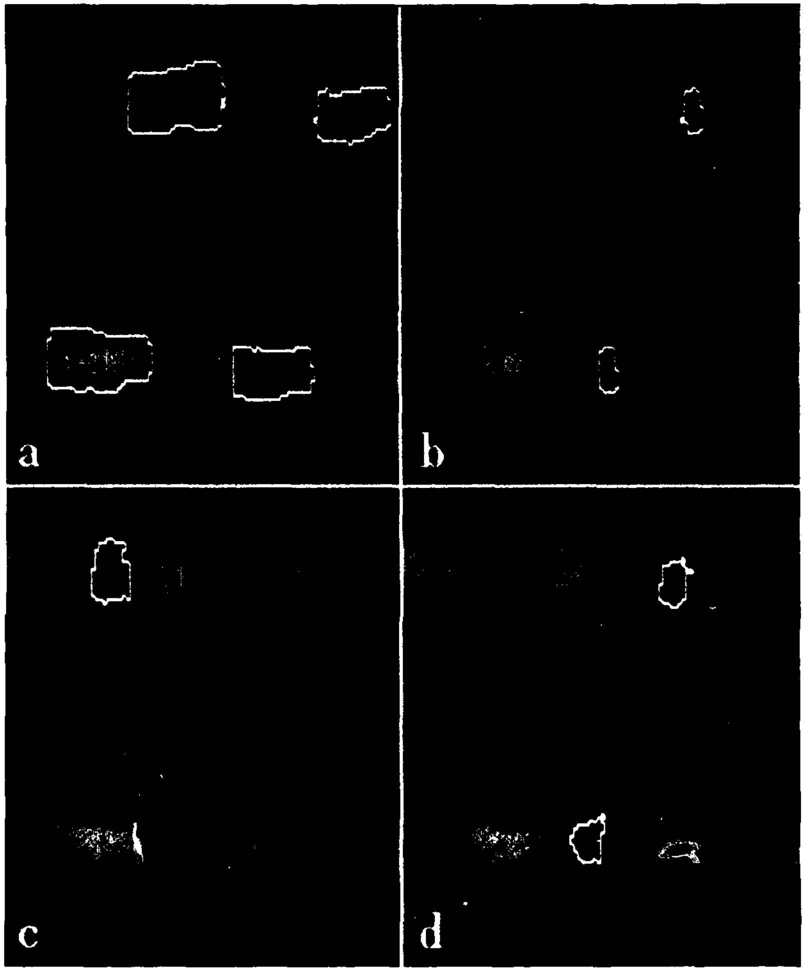


Figure 7. (a) Houses. (b) House-adjunct. (c) House shadows. (d) House-adjunct shadows.

3.3. Experimental Results

The current version of the system has been tested on several images. Figures 6–11 show some results obtained for small parts of two suburban neighborhoods.

Figures 6a and 6b show original and SNF-enhanced images of the first neighborhood, while Figures 6c and 6d show the instances of roads, houses, house-adjuncts, and their shadows found first by search among the high-contrast components, then accumulated during subsequent search among low-contrast components. Black borders indicate the individual components of the level of segmentation, which constitute a basis for search by merging; white borders indicate actually found instances of different types. Figures 7a–d and 8 show instances found by type, with their outlines overlaid on the original image, to give some feeling for difficulty and accuracv.

Because our Prolog image theory is rudimentary, parts of the image haven't been fully interpreted. For

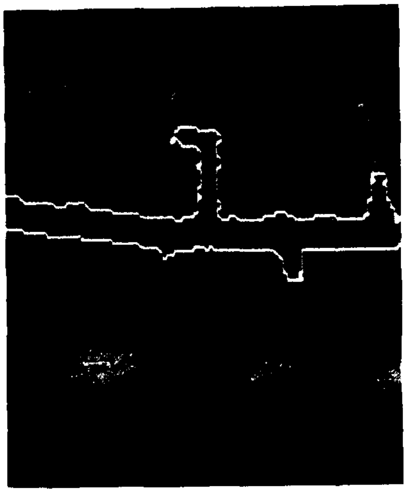


Figure 8. Road complex.

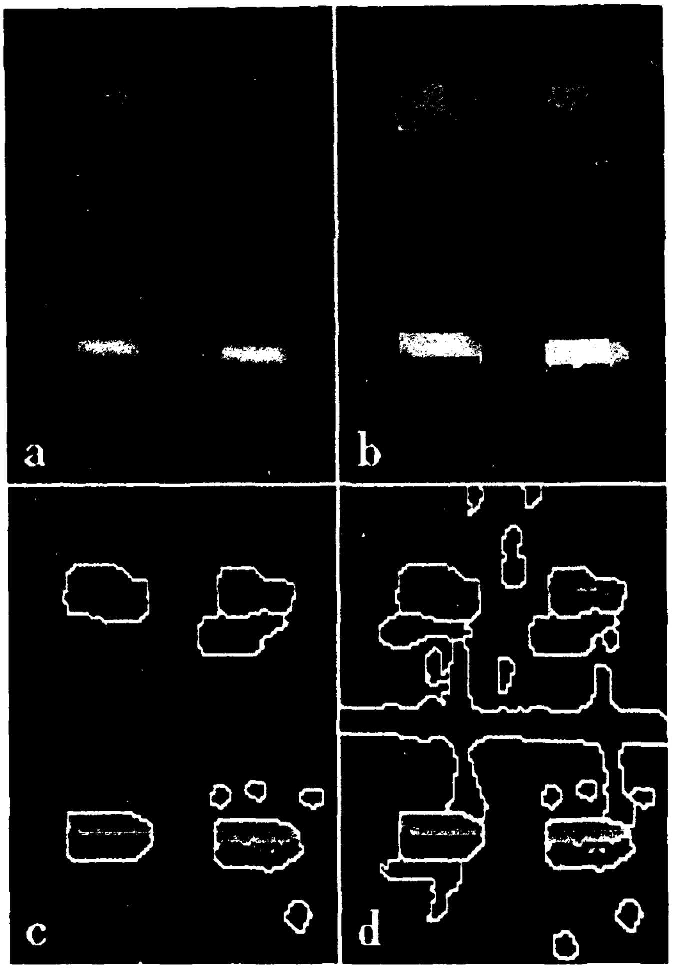


Figure 9. (a) Original image. (b) SNF enhanced image. (c) High-contrast search. (d) Medium-contrast search.

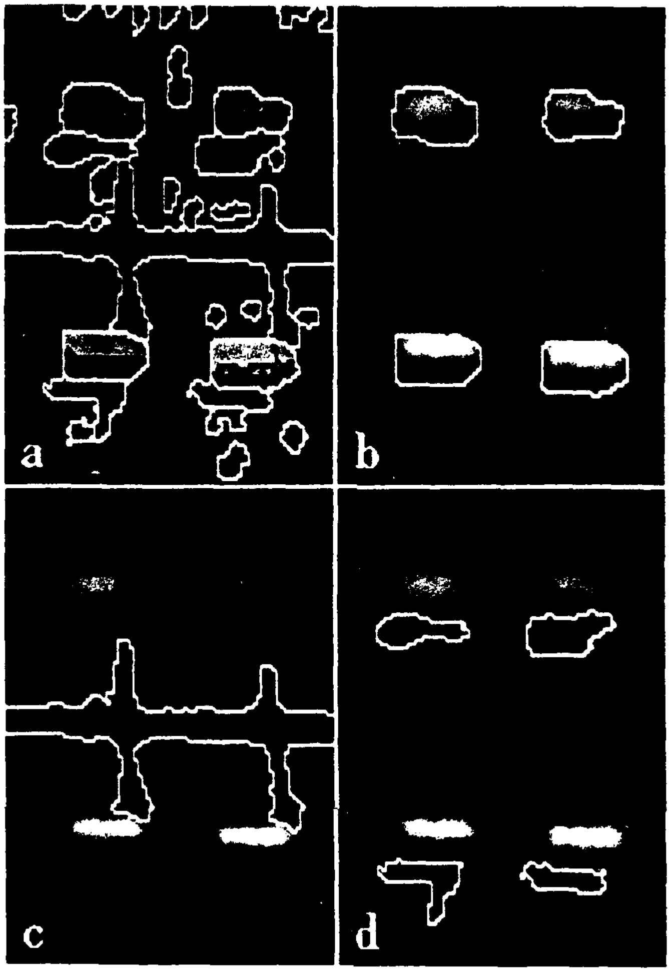


Figure 10. (a) Low-contrast search. (b) Houses. (c) Road complex. (d) House shadows.

example, we know that the house-adjuncts are not garages since they are not approached by driveways. But we need a theory of shading to properly interpret the whole structures of these houses. (Note that the sun direction is to the right, also that the shadows of the adjuncts are small compared to those of simple houses.) Shading and shadows are very complex, being determined by sun direction and the geometry of shaded and overshadowing objects. This points to the need for general "theories" of shading and overshadowing, and of occlusion, even in 2D image interpretation. This is especially important in reasoning abou' missing or conflicting visual evidence, for diagnosis of errors and "virtual" delineation of objects whose evidence is weak.

Figures 9-11 show four houses, their shadows, trees, and the road; apparently there are two vehicles, one on a driveway, the other near the street- the first much larger. Note that the enhanced image preserves the roof structure very well. Note also that the delineation of the shadows of the houses give useful information about the height and structure of the houses: one house has two stories, while another is split-level, with the short side toward the vehicle, perhaps indicating an attached garage.

There is a considerable amount of visual information even in poor quality aerial images such as these (6 bits, 40 gray levels, with poor contrast and blur). We are able to detect and delineate objects which have a figure-ground contrast of only one or two gray levels, and about which we have to reason in order to identify them, however uncertainly.

3.4. Conclusion

The first version of the PPL system has been implemented fully and the results so far have been very encouraging. The performance of the system can be improved by strengthening our image theory by adding new types of objects and relations; this will require more powerful and computationally expensive image analysis.

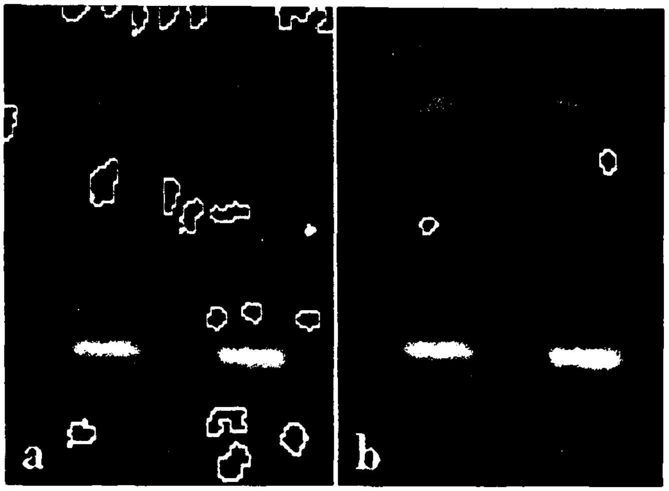


Figure 11. (a) Trees. (b) Two vehicles (near road and on driveway).

We also intend to pursue five areas of research: (i) Development of a parallel logic-programming system for implementation of PPL and other complex programs for image interpretation, probably using our 128 node Butterfly multi-processor. (ii) Investigation of region-based and feature-based stereo analysis and its relation to image interpretation. (iii) Introduction of image theories for shaded, overshadowed, and occluded objects. (iv) Implementation of object-specific image resegmentation, especially to detect and delineate small objects and thin structures. (v) Research on more powerful, heuristic search procedures based on edge and region compatibility relations among components.

REFERENCES

Bratko, I. (1986) *Prolog Programming for Artificial Intelligence*, Addison-Wesley, Reading.

Harwood, D.A., Margalit, A., Prasannappa, R., and Davis, L.S., "Image Enhancement by Symmetric Neighborhood Filters", Center for Automation Research, University of Maryland, technical report in preparation.

Maier, D. and Warren, D.S. (1988) *Computing with Logic*, Benjamin/Cummings, Menlo Park.

McKeown, D.M. Jr., Harvey, W.M., and McDermott, J. (1985) "Rule-based Interpretation of Aerial Imagery", *IEEE Trans. PAMI*, **7**, 570-585.

Nazif, A.M. and Levine, M.D. (1984) "Low-level Image Segmentation: An Expert System", *IEEE Trans. PAMI*, **6**, 555-577.

Selfridge, P.G. (1982) "Reasoning about Success and Failure in Aerial Image Understanding", Ph.D. Thesis, University of Rochester.

van Caneghan, M. and Warren, D.H.D., eds. (1986) *Logic Programming and Its Applications*, Ablex Publishing Co.

AN INTRODUCTION TO GENERALIZED STEREO TECHNIQUES

Lawrence Brill Wolff[1]
Computer Science Department
Columbia University
New York, N.Y. 10027

ABSTRACT

Developed herein is a formal theory for stereo vision which unifies existing stereo methods and predicts a large variety of stereo methods not yet explored. The notion of "stereo" is defined axiomatically using terms which are both general and precise giving stereo vision a broader and more rigorous foundation. The variations in imaging geometry between successive images used in parallax stereo and conventional photometric stereo techniques are extended to stereo techniques which involve variations of arbitrary sets of physical imaging parameters. Physical measurement of visual object features is defined in terms of solution loci in feature space arising from constraint equations that model the physical laws that relate the object feature to specific image features. Ambiguity in physical measurement results from a solution locus which is a subset of feature space larger than a single measurement point. Stereo methods attempt to optimally reduce ambiguity of physical measurement by intersecting solution loci obtained from successive images. A number of examples of generalized stereo techniques are presented. This new conception of stereo vision offers a new perspective on many areas of computer vision including areas that have not been previously associated with stereo vision (e.g. color imagery).

As the central focus of generalized stereo vision methods is on measurement ambiguity mathematical developments are presented that characterize the "size" of measurement ambiguity as well as the conditions under which disambiguation of a solution locus takes place. The *dimension of measurement ambiguity* at a solution point is defined using the structure of a differentiable manifold and an upper bound is established using the implicit function theorem. The symmetry group of bijective transformations of feature space into itself, which leave a solution locus invariant, are used to analyze the ambiguity of a solution locus. A purely group theoretic characterization of the conditions under which measurement disambiguation takes place is given.

1 INTRODUCTION

It is felt that the existing set of vision techniques which are associated with the notion of "stereo" reflect a conceptual basis for stereo vision which is much too limited. For the most part stereo vision implies the traditional techniques of parallax stereo which measures the depth, or more generally, the 3D coordinate position of a point on an object surface, from the intersection of distinct rays of projection onto image planes. Clever variations on parallax stereo such as with intensity gradient light striping in [Schwartz 1983] and with one-eyed stereo in [Strat and Fischler 1985] ultimately must rely on conceptually equivalent triangulation methods to measure the position of points.

A set of techniques to measure local surface orientation using a Lambertian reflectance map for material surfaces was introduced in [Woodham 1978] and named *photometric stereo*. Although completely distinct from the techniques of parallax stereo, the techniques of photometric stereo also rely upon a sequence of successive images to compute local surface orientation. Unlike parallax stereo, the techniques of photometric stereo keep the image plane static and vary the incident orientation of a single light source between successive images. Also, in photometric stereo image irradiance at a corresponding pixel is extracted rather than 2D image coordinate position as in parallax stereo.

Another set of stereo techniques to measure local surface orientation using a more physically accurate reflectance map was introduced in [Wolff 1987a]. The single incident light source and the image plane are held static between successive images. Only variations in the incident wavelength and/or polarization of the light source are made between successive images. As in photometric stereo the image irradiance at the same specified pixel in each successive image produces a distinct equireflectance curve in gradient space. The intersection point(s) for all equireflectance curves is the measurement of local surface orientation.

The existence of photometric stereo as well as spectral and polarization stereo techniques which fall in the same category of techniques as parallax stereo under the intuitive idea of "stereo" suggests a much broader foundation for stereo vision. But then how can a stereo vision technique be formally defined or characterized ? Is there a cohesive framework within which to describe *all existing stereo techniques ?*

Informally, a stereo vision technique performs a quantitative physical measurement of some kind of visual object feature using a sequence of successive images such that between each successive image at least one parameter of the imaging system is varied. The visual object feature is not directly observed, rather, only image functions are empirically ascertained and then certain well defined operations are performed on each image function. These operations on image functions are simple for the cases of parallax and photometric stereo and spectral and polarization stereo. Assuming that correspondence has already been solved for a given parallax stereo technique measuring the 3D position of an object point, the image operation performed extracts from each successive image the 2D image coordinate position of the correspond-

[1] This work was supported in part by ARPA grant #N00039-84-C-0165. This work was done in part while the author was in the *Machine Perception Research* Department at AT&T Bell Laboratories, Holmdel, N.J. . This work was supported in part by an IBM Graduate Fellowship Award.

ing pixel. For photometric stereo and spectral and polarization stereo the image irradiance is extracted at the corresponding pixel. Other stereo techniques have been developed which require more elaborate image operations.

After extracting appropriate numerical values using an image operation, the physical measurement of a visual object feature requires the solution of a system of one or more equations which relate these extracted numerical values to the visual object feature. This system of equations which arises for each successive image can be interpreted as the camera model with respect to the given object feature. In the case of traditional parallax stereo the system of equations arising from each image are the two perspective projection equations relating the corresponding 2D image coordinate position to the 3D world coordinate position visual feature. For photometric stereo and spectral and polarization stereo techniques a single image irradiance equation arises for each successive image relating the image irradiance at the corresponding pixel to the reflected scene radiance at the point on the object at which local surface orientation is to be measured.

For a given stereo vision technique the solution(s) of the system of equations for each successive image can be interpreted as a solution locus of points in a feature vector space which is spanned by the physically independent parameters used to represent the visual object feature to be measured. A set of object or imaging parameters are said to be *physically independent* if the physical variation of the value for any one parameter does not effect the values for any of the other parameters. Valid *measurement points* exist only at the mutual intersection of all solution loci obtained from all successive images. That is, a point on a given solution locus is a measurement point with respect to the sequence of successive images taken for a stereo vision technique if and only if this point is "corroborated" as being on the solution loci for all successive images with respect to variations that are performed on a preselected set of imaging parameters. The essence of physical measurement by stereo vision techniques is to isolate valid measurement points from the *disambiguation* of intersecting solution loci.

The resulting solution loci from each successive image for traditional parallax stereo are lines in a 3-dimensional Euclidean feature space of world coordinate position points. Photometric stereo and spectral and polarization stereo techniques generate solution loci which are one dimensional equireflectance curves in a feature space which, for example, can be 2-dimensional gradient space representation for surface orientation. From this more general perspective, it is theoretically possible for there to exist stereo vision techniques which isolate valid measurement points from the intersection of multi-dimensional solution loci in a higher dimensional feature space. In fact a stereo technique in [Wolff 1987b], and summarized in this paper, to measure surface curvature intersects 2 dimensional solution loci in a 5 dimensional feature space. The mathematics of disambiguation in this case can be rather abstract and complicated, and establishing useful tools to quantify the *size* of measurement ambiguity as well as the conditions under which disambiguation takes place occupies much of the development in this paper.

In summary, four distinct quantities are seen to characterize a stereo vision technique at the level of unifying all existing stereo techniques. The first is the *visual object feature* to be measured. The second is the *image operation* performed on each successive image extracting appropriate numerical values. The third is a preselected set of *imaging parameters* of which at least one parameter is varied between each successive image. The fourth is the system of *camera modeling equations* which relate the first three quantities together. These four quantities will be formally defined in section 2.

For a given generalized stereo vision technique, the physical measurement produced from the mutual intersection of solution loci in an arbitrary multi-dimensional feature space is said to be *ambiguous* if the intersection consists of more than a single point. The term *ambiguity* does not in any way imply experimental error. Rather the ambiguity of the quantitative measurement of visual object features arises from the nonunique solution to the equations that relates the value of the object feature to functions of the corresponding image pixel values. A solution locus which consists of more than a single point in feature space is considered to be an ambiguous physical measurement since there is an uncertainty in the value of the object feature. This ambiguity of physical measurement exists independent of experimental error even though nonzero experimental error clearly exacerbates the uncertainty of the state of an object feature with a further kind of ambiguity.

Certain *measures of* measurement ambiguity are proposed in this paper assuming that connected components of the intersection of solution loci form *manifolds*. A global measure of measurement ambiguity simply integrates volume elements of appropriate dimensionality over the entire mutual intersection of the solution loci. A local measure of measurement ambiguity utilizes the implicit function theorem applied to the camera model equations for the fourth axiom of generalized stereo to compute an upper bound on the *dimension* of measurement ambiguity at a point. This characterizes the dimension of a local neighborhood of a point on the mutual intersection of solution loci.

Suppose for a given generalized stereo technique that the first image produces the solution locus M_1 in feature space. Let the solution locus resulting from the second image be M_2. The solution locus M_2 is said to *disambiguate* solution locus M_1 if and only if $M_1 \cap M_2 \subsetneq M_1$. This paper presents a theoretical characterization of the conditions under which the solution locus M_1 is disambiguated by another successive image. This is done using the notion of the *symmetry* of a solution locus, precisely expressed in the language of group theory.

A *symmetry* for a solution locus in feature space is a one-to-one, onto transformation of feature space coordinates, called an *automorphism*, that preserves the set of points occupied by the solution locus when it is transformed. For photometric stereo and spectral and polarization stereo the equireflectance curves in gradient space possess a flip symmetry about a specified line. The feature space coordinate transformation that represents this symmetry may move solution locus points around, but the end result is that the transformed solution locus looks exactly the same as the original solution locus. A useful property of symmetries for a solution locus is that, with respect to composition of transformations, they form a group which in turn is a subgroup of the set of all transformations of feature space into itself.

The variation of imaging parameters between successive images ind ıces a coordinate transformation on feature space, which will be called a *stereo automorphism*. If the first image produces solution locus M_1 and the second image produces solution locus M_2, then the stereo coordinate automorphism induced by the variation of imaging parameters between the first and second images must transform the point set M_1 into the point set M_2. Clearly if the corresponding stereo automorphism coincides with a symmetry automorphism of M_1, then $M_1 = M_2$ and no disambiguation takes place. It is shown that in fact disambiguation cannot take place for a much larger set of stereo automorphisms. This paper establishes a theorem that states that disambiguation of M_1 can take place if and only if the stereo automorphism induced by the variation of imaging parameters between successive images is outside the *normalizer subgroup* of the symmetry subgroup of M_1, in the group of all automorphisms of feature space into itself.

2 GENERALIZED STEREO

Before presenting an axiomatic definition of stereo vision in terms of a formalization of the four quantities *visual object feature*, *image operation*, *imaging parameters* and *camera modeling equations*, some development needs to be given towards the formalization of an image operation. Then, after formally defining stereo vision in a general way, a number of stereo vision techniques will be discussed in context with this definition. Many of these stereo vision techniques are new and have only been presented in publications dealing with different phases of this research.

In general an s color image can be represented by an s-tuple image function, I_s, which is considered to be a function from $\mathbb{R}^2$ into the set of real s-tuples, $(c_1, c_2, \ldots, c_s)$, for $s \geq 1$. An image operation upon an s-tuple image function can produce a t-tuple image function for any $t \geq 1$. So for instance a gradient image operation upon a differentiable 1-tuple image function produces a 2-tuple image function of $\mathbb{R}^2$ mapped into real gradient vectors in 2 dimensions. An image operation therefore defines a *functional*[2] from the set of s-tuple image functions into the set of t-tuple image functions. For such an image operation functional, F, acting on the s-tuple image function, I_s, the evaluation of the real t-tuple at the point (i.e. pixel) $p \in \mathbb{R}^2$ *for the corresponding* t-tuple image function is denoted by $F_t(I_s)|_p$. This will be called a *point valued* image operation functional.

A more general image operation functional which can evaluate real t-tuples over arbitrary point regions can be defined using the power set, $P(\mathbb{R}^2)$, of $\mathbb{R}^2$. This type of image operation functional maps s-tuple image functions into functions from $P(\mathbb{R}^2)$ into real t-tuples. An example of such a functional assigns an s-tuple image function, I_s, to a function from $P(\mathbb{R}^2)$ to the 1-tuple area value of each subset of pixel points [3]. Another example of such a functional assigns an s-tuple image function, I_s, to a function from $P(\mathbb{R}^2)$ to the s-tuple which represents the average over each subset of pixel points. Analogous to the definition of notation above, if F acts on the s-tuple image function I_s, the evaluation of the real t-tuple at the region $A \subset \mathbb{R}^2$ for the corresponding t-tuple image function is denoted by $F_t(I_s)|_A$. This will be called a *region valued* image operation functional even though it is understood that the set of all regions include point regions as well.

The formal axiomatic definition of stereo vision states that a stereo vision technique is completely defined by the following four quantities:

- *Visual object feature* to be measured represented as a real n-tuple of physically independent parameters $(x_1, x_2, \ldots, x_n)$.
- A well defined *image operation functional* on the set of s-tuple image functions into the set of t-tuple image functions, $F_t(I_s)|_p$ for point valued, $F_t(I_s)|_A$ for region valued, $s, t \geq 1$.
- A preselected set of physically independent *image system parameters*, $(a_1, a_2, \ldots, a_m)$, at least one of which is varied between each successive image.
- *Camera modeling equations* involving the quantities defined in the first three axioms:

$$\left.\begin{array}{c}\Phi_1(x_1, x_2, \ldots, x_n, a_1^{(k)}, a_2^{(k)}, \ldots, a_m^{(k)}, F_t^{(k)}(I_s)|_A) = 0\\ \Phi_2(x_1, x_2, \ldots, x_n, a_1^{(k)}, a_2^{(k)}, \ldots, a_m^{(k)}, F_t^{(k)}(I_s)|_A) = 0\\ \vdots\\ \Phi_h(x_1, x_2, \ldots, x_n, a_1^{(k)}, a_2^{(k)}, \ldots, a_m^{(k)}, F_t^{(k)}(I_s)|_A) = 0\end{array}\right\} \quad (1)$$

the superscript (k) implying a dependence on each successive image. A point valued image operation functional could of just as well been used.

To distinguish from stereo vision techniques which are related exclusively to parallax stereo techniques, the complete set of stereo techniques which are defined by the four axioms above will be called *generalized stereo* techniques. The real valued n-tuples and m-*tuples* from the first and third axioms above will also be termed object feature and imaging system *states* respectively.

Even more generally, a different image operation functional can be used for each successive image, but particular generalized stereo methods of this kind will not be presented here. Examples of generalized stereo techniques presented below have a constant number of equations, $h \geq 1$, produced from each successive image. This definition of generalized stereo excludes any auxiliary constraints provided by scene modeling, scene conditioning (e.g. light striping) or any *a priori* assumptions.

For $(l + 1)$ successive images there will be $h(l + 1)$ equations in all. As implied by equations 1 the physical state of the object feature to be measured is assumed to remain invariant while the physical state of the imaging system is assumed to be altered between each successive image. Let M_0 be the solution locus obtained from the first image and let M_i be the solution locus obtained from the $(i + 1)st$ successive image, for $1 \leq i \leq l$. Then the solution locus M_0 is said to be *disambiguated* with respect to the stereo sequence of (l+1) successive images if and only if

$$\cap_{i=0}^{l} M_i \subsetneq M_0 .$$

The "size" of the resulting solution locus and the conditions under which a *solution locus can be disambiguated are* topics discussed in sections 3 and 4.

The definition of generalized stereo not only serves as a precise unifying framework for existing "stereo" techniques but predicts a vast variety of techniques which have not yet been explored. Some of these new generalized stereo techniques may be of significant practical and even biological importance. Presented now are a number of stereo techniques some of which are familiar and some of which are new.

[2] The term *functional* implies a function from one set of functions to another set of functions.

[3] The area value can be precisely defined as being the two dimensional Lebesgue area measure.

EXAMPLE 1 (PARALLAX STEREO)

This is the original stereo vision algorithm that has been in existence for thousands of years and is the algorithm that is usually implied when the term "stereo" is mentioned [Ballard and Brown 1982]. As discussed in the introduction, the visual object feature to be measured is the depth, or more generally, the three dimensional world spatial position of a point on an object demarcated by some prominence like an edge marking. A pixel valued 2-tuple image operation functional is used which maps any image function into a function which maps each pixel in $\mathbb{R}^2$ (the image array of pixels) into the 2-tuple representing its two dimensional location using a given image coordinate system. The 2-tuple valued image operation functional is evaluated at the pixel corresponding to the manifestation of the object point in the image and assuming the projection mapping depicted in figure 1, two perspective equations arise for each successive image. The preselected physical parameters to be varied between successive images are the three parameters specifying the world spatial position of the focal point which are represented by $(a_1^{(k)}, a_2^{(k)}, a_3^{(k)})$ for the k^{th} successive image, being consistent with the notation in equations (1). If the image coordinates of the object point in the k^{th} successive image are $(u^{(k)}, v^{(k)})$, the focal length is always f and assuming no vergence, then the resulting equations are

$$\left.\begin{array}{l} \frac{a_1^{(k)}-x_1}{u^{(k)}} - \frac{a_3^{(k)}-x_3}{f} = 0 \\ \frac{a_2^{(k)}-x_2}{v^{(k)}} - \frac{a_3^{(k)}-x_3}{f} = 0 \end{array}\right\}$$

which not surprisingly is the intersection of two planes in the three dimensional feature space of world position coordinates. In fact two successive images overconstrain the point solution with the intersection of four planes.

Auxiliary scene conditioning such as light striping [Schwartz 1983] adds an auxiliary planar solution locus in feature space to the intersection of two planes produced from a single image to get a unique solution point. Another monocular stereo technique is employed by [Strat and Fischler 1985] who generate a second hypothesized *virtual image* from an appropriate scene model. This also adds an auxiliary solution locus in the world feature space to obtain a unique solution point from a single image. These methods are not considered here to be *pure* stereo methods because they employ auxiliary assumptions or constraints.

EXAMPLE 2 (PHOTOMETRIC STEREO)

This is a stereo technique first proposed in [Woodham 1978] to determine the shapes of material surfaces assuming a Lambertian reflectance map. Related stereo techniques are used in [Silver 1980] and [Ikeuchi 1987]. The visual object feature to be measured is local surface orientation and the image operation functional used is simply the identity mapping from 1-tuple image irradiance functions on $\mathbb{R}^2$ to the same 1-tuple image irradiance functions on $\mathbb{R}^2$. That is, all that needs to be evaluated is the image irradiance value $I^{(k)}$ at the corresponding pixel *in the* k^{th} successive image. The physical state of the imaging system to be altered is the incident light orientation represented by the 2-tuple $(a_1^{(k)}, a_2^{(k)})$ denoting gradient space coordinates (p, q) respectively. The physical state of the surface orientation (x_1, x_2) will be solved for in a two dimensional feature space which in gradient space coordinates yields the following equation, assuming a Lambertian reflectance map, for the k^{th} successive image

$$\frac{\rho(\ 1 + x_1 a_1^{(k)} + x_2 a_2^{(k)}\ )}{\sqrt{1 + (x_1)^2 + (x_2)^2}\sqrt{1 + (a_1^{(k)})^2 + (a_2^{(k)})^2}} - I^{(k)} = 0 \qquad (2)$$

with albedo ρ. The resulting solution locus from each successive image is commonly referred to as an equireflectance curve.

EXAMPLE 3 (TEXEL STEREO)

Even though stereo techniques for the determination of shape from texture have never been formally specified it is a small conceptual jump from the works of [Kender 1980], [Ohta et al. 1980] and [Aloimonos and Swain 1985], noting that texel area elements exhibit the same behavior as the reflected intensity from a Lambertian surface, that a texel stereo technique equivalent to that of photometric stereo could be developed by modifying the image operation functional used and the preselected set of physical parameters to be varied. Of course the visual object feature to be measured is local surface orientation. The region valued 1-tuple image operation functional to use was mentioned earlier and assigns image functions to the function that assigns every subset of pixels the area covered by that subset. The physical parameters to vary specify viewer orientation and is represented in gradient space coordinates $(a_1^{(k)}, a_2^{(k)})$ for the k^{th} successive image. Assuming the actual surface area of the texel to take the place of the albedo ρ and the evaluation of the 1-tuple image operation functional at the appropriate region of pixels is $A^{(k)}$, the equation that arises in gradient feature space for the k^{th} successive image is the same as for equation 2 with $A^{(k)}$ substituted for $I^{(k)}$.

EXAMPLE 4 (SPECTRAL AND POLARIZATION STEREO)

The stereo techniques presented so far alter the physical state of the imaging system solely with respect to some aspect of the imaging geometry. First proposed in [Wolff 1986] and also reported in [Wolff 1987a] are stereo techniques that measure local surface orientation of material surfaces, using an accurate reflectance model, by only varying the incident wavelength and/or linear polarization of a point light source while leaving the imaging geometry completely invariant. The image operation functional used is the same as for photometric stereo, namely image irradiance values are measured in successive images.

The reflectance model that is used was proposed in [Torrance and Sparrow 1967] and is applicable to a wide variety of isotropically *rough homogeneous dielectric and conducting surfaces*. The form of the Torrance-Sparrow reflectance function is

$$\rho\{\ g I_0 R_s + I_0 R_d\ \}$$

where the terms R_s and R_d are the functions for the specular and diffuse components of reflection respectively, the term I_0 represents the incident radiance of the light source, ρ is the albedo and g is the amplitude proportion of the specular reflectance relative to the diffuse reflectance function. The detailed forms for the functions R_s and R_d can be found in [Wolff 1987a]. To be consistent with the unified notation in equations (1) the Torrance-Sparrow reflectance function will be represented as

$$\rho\{\ g I_0[\ a_1 F_\perp(\Psi', \eta(a_2)) + (1 - a_1) F_\parallel(\Psi', \eta(a_2))\ ] r_s(x_1, x_2, p_s, q_s) \\ + I_0 R_d(x_1, x_2, p_s, q_s)\ \} \qquad (3)$$

with a_1 varying between 0 and 1 inclusive being the physical parameter which specifies incident linear polarization, a_2 being the incident wavelength and $(p_s, q_s, -1)$ being the incident light orientation in gradient space coordinates. So for instance a RGB color image could be interpreted as a stereo image with respect to the superimposition of three successive images taken for a_2 set to a red, green and blue wavelength respectively.

The exact forms for $r_s(x_1, x_2, p_s, q_s)$ and $R_d(x_1, x_2, p_s, q_s)$ are extraneous to the discussion here. What is important to note is that the expression for the reflected scene radiance in 3 is non-linearly dependent upon the physical parameters a_1 and a_2. The *Fresnel reflection coefficients* $F_\perp(\Psi', \eta(a_2))$ and $F_\|(\Psi', \eta(a_2))$ for perpendicular and parallel linearly polarized light respectively are dependent on the term $\Psi' \equiv \Psi'(p_s, q_s)$ which is the angle of incidence on a planar microfacet and equal to half the phase angle. For pure spectral and polarization stereo methods this angle remains invariant as the incident light orientation is unchanged. The Fresnel reflection coefficients are also dependent on the term $\eta = n - i\kappa$ which is the complex index of refraction for the material surface and its components are in turn dependent on the wavelength parameter a_2 according to

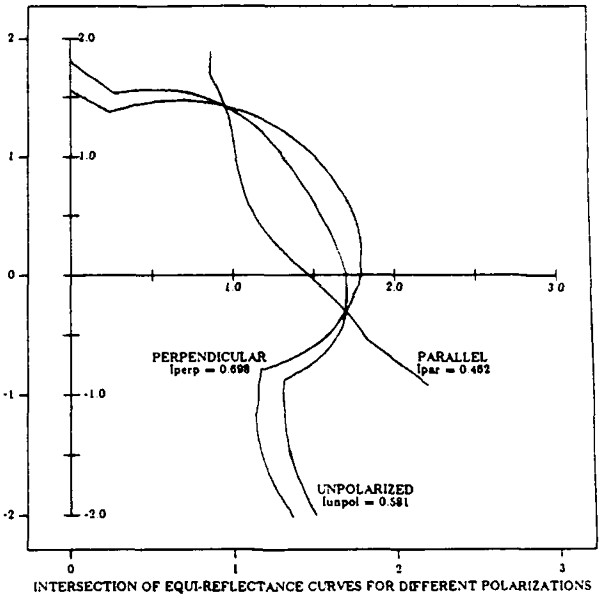
$$\begin{aligned} n^2 &= \tfrac{\nu\gamma c_0^2}{2}(1 + [1 + (\tfrac{a_2}{2\pi c_0 r_e \gamma})^2]^{1/2}) \\ \kappa^2 &= \tfrac{\nu\gamma c_0^2}{2}(-1 + [1 + (\tfrac{a_2}{2\pi c_0 r_e \gamma})^2]^{1/2}) \end{aligned} \qquad (4)$$

where c_0 is the speed of light in vacuo, r_e is the electrical resistivity of the surface material, and ν and γ are the electrical permitivity and the magnetic permeability of the surface material respectively. If the image irradiance value measured for the k^{th} successive image is $I^{(k)}$ the following equation arises in gradient space solving for the orientation feature state (x_1, x_2)

$$\rho\{\ gI_0[\ a_1^{(k)} F_\perp(\Psi', \eta(a_2^{(k)})) + (1 - a_1^{(k)}) F_\|(\Psi', \eta(a_2^{(k)}))\ ] r_s(x_1, x_2, p_s, q_s)$$
$$+ I_0 R_d(x_1, x_2, p_s, q_s)\ \} - I^{(k)} = 0 \qquad (5)$$

Figure 1 shows the intersection of equireflectance curves solving for surface orientation by varying incident linear polarization. It was noted in [Wolff 1987a] for surface orientations not lying on the line in gradient space passing through the origin and having slope p_s/q_s that a two point physical measurement ambiguity will always result regardless of how many successive images are taken varying the image system state represented by $(a_1^{(k)}, a_2^{(k)})$ specifying incident linear polarization and wavelength. The reason for this is discussed further in section 4. It is noted that one solution to obtain unique measurement is to include one of the parameters p_s or q_s specifying incident light source orientation in the preselected set of physical parameters to alter the image system state. Another method which does not alter the orientation of the light source is to instead include a single physical parameter in the preselected set specifying the solid angular subtending area of a preset asymmetric shape of an adjustable diaphram in front of an extended light source.

Equations 4 imply that other physical parameters independent of imaging geometry and incident wavelength and linear polarization can be selected to obtain a measurement of local surface orientation. The variables r_e, ν and γ are dependent on temperature, external electric and magnetic fields and tensile stress on the material. It is possible to have stereo techniques measuring surface orientation by preselecting physical parameters representing the alteration of the state of the image system with respect to these degrees of freedom.



COMBINED SPECULAR AND DIFFUSE REFLECTANCE MAP
REFLECTANCE FUNCTION 0.5*(2.0*Rs + Rd)
SOURCE VECTOR ORIENTATION Ps=7.0 Qs=3.0
MAGNESIUM OXIDE N=1.77 K=0.00 (365 NM)
RMS SLOPE VALUE= 2.0

Figure 1

In [Wolff 1987c] another polarization stereo technique is presented for more robust practical circumstances. Surface orientation is computed from knowledge of the incident state of polarization and the *polarization image* which is derived from an appropriate stereo sequence of reflected radiance detected by the sensor for different orientations of a linear polarizer placed in front of the sensor.

EXAMPLE 5 (SURFACE CURVATURE FROM STEREO)

With knowledge of the reflectance map of a material surface, measurement of the Gaussian curvature at a point on the surface can be easily achieved from the two previous examples of stereo methods by first obtaining the normal map to the surface over a region local to the point and then computing the change in the surface normals at the point in two linear independent directions. In effect the Gaussian curvature is determined from local contextual knowledge with respect to surface orientation in the vicinity of the point on the surface. In [Wolff 1987b] a photometric method is proposed to determine the Gaussian curvature at a point on a material surface using measurements of the gradient of the reflected scene radiance between successive images. The contextual knowledge used is in the image itself to compute the image gradient from neighboring pixels. In this way neighboring surface orientations about the point on the material surface are not required to be computed.

The advantages of determining the Gaussian curvature from this technique with respect to measurement accuracy are discussed in [Wolff 1988]. Of interest here is the observation that this technique is another example of a generalized stereo method. Parameterizing the object surface with respect to height above the image plane as $(\ u, v, f(u, v)\ )$ using image coordinates (u, v).

the object feature to be measured is represented by the 5-tuple state

$$(\partial f/\partial u, \partial f/\partial v, \partial^2 f/\partial u^2, \partial^2 f/\partial v^2, \partial^2 f/\partial u \partial v)$$

which in this example will correspond to the point $(x_1, x_2, x_3, x_4, x_5)$ in 5 dimensional feature space. The last three components of the 5-tuple can be recognized as the three parameters determining the *surface hessian* matrix. However from a more general point of view the above 5-tuple completely specifies both the first and second order rates of change of the surface height function $f(u, v)$ which are necessary to derive the Gaussian curvature from the Gauss-Weingarten equations [DoCarmo 1976].

Evaluated in each successive image is a point valued 3-tuple image operation functional mapping the image function I(u,v) of coordinates (u,v) in $\mathbb{R}^2$ to the function mapping each point (i.e. pixel) in $\mathbb{R}^2$ into the 3-tuple $(I, \partial I/\partial u, \partial I/\partial v)$. From [Wolff 1987b], the three equations that arise for the k^{th} successive image are

$$\left.\begin{array}{c} R(x_1, x_2, a_1^{(k)}, a_2^{(k)}, \ldots, a_m^{(k)}) - I^{(k)} = 0 \\ \frac{\partial R(x_1, x_2, a_1^{(k)}, a_2^{(k)}, \ldots, a_m^{(k)})}{\partial x_1} x_3 + \frac{\partial R(x_1, x_2, a_1^{(k)}, a_2^{(k)}, \ldots, a_m^{(k)})}{\partial x_2} x_5 - (\frac{\partial I}{\partial u})^{(k)} = 0 \\ \frac{\partial R(x_1, x_2, a_1^{(k)}, a_2^{(k)}, \ldots, a_m^{(k)})}{\partial x_1} x_5 + \frac{\partial R(x_1, x_2, a_1^{(k)}, a_2^{(k)}, \ldots, a_m^{(k)})}{\partial x_2} x_4 - (\frac{\partial I}{\partial v})^{(k)} = 0 \end{array}\right\} \tag{6}$$

with respect to the reflectance function $R(x_1, x_2, a_1, a_2, \ldots, a_m)$ which is shown to be dependent upon an arbitrary preselected set of parameters $(a_1, a_2, \ldots, a_m)$ for the image system state implying the use of a variety of preselected parameter sets including those used in the stereo methods presented in the last two examples.

This is a good example of generalized stereo because it illustrates just how complicated the sets of solution loci can get in a higher dimensional feature space. Assuming sufficient differentiablity of the reflectance function and a complete normed 5 dimensional vector space structure on the set of 5-tuples $(x_1, x_2, \ldots, x_5)$, each equation in the system of equations 6 determines a 4-dimensional *sub-manifold* solution locus. The sub-manifold for the first equation in this set can be mentally visualized as the "sweeping out" of a 1 dimensional equireflectance curve embedded in the 2 dimensional subspace spanned by x_1 and x_2 (i.e. gradient space), in the three additional dimensions spanned by x_3, x_4 and x_5 in the directions of their respective axes. The last two equations in this set can be visualized as the "sweeping out" of 2 dimensional planes embedded in the three dimensional subspace spanned by x_3, x_4 and x_5, whose coefficients are determined by the partial derivatives of the reflectance function with respect to x_1 and x_2. This time the "sweeping out" of the planes as x_1 and x_2 vary could be *very complicated* as the *partial* derivatives of the reflectance function is usually dependent on x_1 and x_2. Therefore the planes undergo "turns" and "twists" as they are being swept out.

Even more complicated is visualizing the solution locus that is determined simultaneously from the complete system of equations 6. Specifying this solution locus is important for characterizing the measurement ambiguity associated with determining the 5-tuple feature state from a single image. An important characteristic of the solution locus at a point is its *dimension* which will be explained in more detail in section 3. Intuitively the higher the dimension of the solution locus at a point, the more ambiguity there is inherent to the feature state measurement. As will be seen in section 3 the *implicit function theorem* provides an efficient algorithm for determining an upper bound on the dimen: ion of a solution locus at a point determined by a general set of equations as shown in equations 1. It turns out for a Lambertian or a Torrance-Sparrow reflectance function that equations 6 almost always determine a 2 dimensional measurement submanifold in 5 dimensional feature space.

3 GLOBAL AND LOCAL MEASURES OF MEASUREMENT AMBIGUITY

In section 2 generalized stereo was presented as a technique for disambiguating the measurement of the state of a visual object feature by adding constraints to a solution locus in feature space. Additional constraints are obtained from the physical equations relating the evaluation of an image operation functional to the object feature state through physical parameters which alter the state of the image system between successive images. Hopefully with each successive image the "size" of the solution locus in feature space gets reduced. But exactly what does the term "size" mean ?

It is assumed for the rest of this section that the set of feature state n-tuples $(x_1, x_2, \ldots, x_n)$ have a Euclidean norm defined in a natural way so that feature space is identical to E^n. No generality is lost in the development to follow since E^n in an abstract sense is a manifold unto itself and any other complete normed n-dimensional vector space structure would generate a *locally diffeomorphic* manifold.

3.1 A Global Measure

Ideally a generalized stereo technique tries to achieve a unique point state measurement of an object feature. This may be possible using some preselected sets of physical imaging parameters and not with others. If the solution locus is not unique the measurement ambiguity may consist of a discrete number of points. In worse cases the solution locus may be a surface or the union of many disconnected component surfaces of many dimensions.

In the case of a discrete number of points, the cardinality of this point set is clearly a good characterization of the "size" of the resulting measurement ambiguity. In the case of a solution locus comprised of one or many multidimensional surfaces, the integration of infinitesimal volume elements[4] of appropriate dimensionality over the entire extent of the solution locus characterizes the "size". Using this definition of "size", a solution has finite "size" if and only if the solution locus is compact since solution loci are closed subsets of E^n. Assume that either by trial and error or by some more intelligent algorithm that the "size" of the solution locus resulting from a sequence of successive images has been minimized, hopefully to some finite value corresponding to a compact region of feature space. Call this a *preliminary global bounding*.

3.2 A Local Measure: The Dimension Of Measurement Ambiguity At A Point

After a preliminary global bounding has taken place, the next step in reducing measurement ambiguity is to reduce the dimension at all points of the resulting solution locus within the preliminary global bounding region. The dimensionality of the surface in the local vicinity of a point on the solution locus is considered to be a local measure of the "size" of the measurement ambiguity at that point. This subsection is concerned with evaluating the dimension

[4] In general this is the integration over *singular p-chains*

of solution loci at individual points as a characterization of the "size" of measurement ambiguity.

To define the *dimension* of a solution locus at a point in feature space the concept of a $C^{(q)}$ *manifold of dimension* r in n-dimensional Euclidean space E^n is introduced.

DEFINITION: *A nonempty set $M \subset E^n$ is a $\underline{C^{(q)}\text{ manifold of dimension } r}$. $1 \leq r < n$, if M has the property that for every $x_0 \in M$ there exists an open set $U \subset E^n$ containing x_0 and an open set $V \subset E^r$ such that*

(i) there exists a one-to-one function $\phi(y) = (\phi_1, \phi_2, \ldots, \phi_n)$ for $\phi: V \rightarrow U \cap M$ which is $C^{(q)}$ differentiable meaning that the function produced by $q \geq 1$ differentiations w.r.t. any $y_1, y_2, \ldots, y_r$ for each ϕ_i exists and is continuous.

(ii) for all $y \in V$ the Jacobian $\mathbf{J}\phi(y)$ is one-to-one.

A $C^{(q)}$ manifold of dimension r will simply be referred to as an *r-manifold*. Intuitively from a topological point of view an r-manifold is locally indistinguishable from E^r in a sufficiently small region about any point. Condition (i) induces a local coordinatization in an open neighborhood about any point on the r-manifold M with respect to the r-dimensional coordinates of E^r. Condition (ii) assures the existence of an r-dimensional *tangent space* at every point on M. It is clear from condition (ii) that the determinant of $\mathbf{J}\phi(y)$ is nonzero for all $y \in V$.

The *dimension of measurement ambiguity at a point x on a solution locus M in E^n* is an integer r, such that $1 \leq r < n$, if there exists an open set $W \subset E^n$ containing x such that $W \cap M$ is an r-manifold in E^n. Indeed this implies that the dimension of measurement ambiguity may be undefined at some points. The dimension of measurement ambiguity for an isolated point on a solution locus is defined to be zero.

In E^3, for instance, the dimension of measurement ambiguity for a point satisfying the equation of a plane or a sphere is 2 as the respective solution loci are 2-manifolds. Consider the equation

$$(x_1)^2 + (x_2)^2 - (x_3)^2 = 0$$

with a solution locus of two cones that meet head on. The dimension of measurement ambiguity at the point of contact is undefined but is of dimension 2 for any other point on the solution locus. A meaningful definition for the dimension of measurement ambiguity at such singularity points is saved for future work.

The implicit function theorem can be used to determine the dimension of measurement ambiguity at a point on a solution locus determined by the general system of equations 1 in section 2. Assuming that each of the Φ_i are of class $C^{(q)}$, the system of equations 1 determines a class $C^{(q)}$ function $\Phi = (\Phi_1, \Phi_2, \ldots, \Phi_{n-r})$ from E^n into E^{n-r} where $i_k = n - r$, $1 \leq r < n$. The solution locus is therefore represented by $M = \{x \in E^n : \Phi(x) = 0\}$. The implicit function theorem effectively states that if the Jacobian

$$\mathbf{J}\Phi = \begin{pmatrix} \partial\Phi_1/\partial x_1 & \partial\Phi_1/\partial x_2 & \cdots & \partial\Phi_1/\partial x_n \\ \partial\Phi_2/\partial x_1 & \partial\Phi_2/\partial x_2 & \cdots & \partial\Phi_2/\partial x_n \\ \vdots & \vdots & \vdots & \vdots \\ \partial\Phi_{n-r}/\partial x_1 & \partial\Phi_{n-r}/\partial x_2 & \cdots & \partial\Phi_{n-r}/\partial x_n \end{pmatrix}$$

evaluated at a solution point $x_0 \in M$ has rank (n-r) then there exists an open set $W \subset E^n$ containing x_0 such that $W \cap M$ is an r-manifold. That is the dimension of the measurement ambiguity at x_0 is r.

Without loss of generality arrange the functions Φ_i so that the last (n-r) column vectors in $\mathbf{J}\Phi$ are linearly independent and form the (n-r)x(n-r) matrix $\tilde{\mathbf{J}}\Phi$ out of these vectors. Clearly the determinant $|\tilde{\mathbf{J}}\Phi| \neq 0$. Let

$$\hat{x} = (x_1, x_2, \ldots, x_r), \qquad \hat{x}_0 = ((x_0)_1, (x_0)_2, \ldots, (x_0)_r)$$

denote the vectors by taking the first r components of x and x_0. The following is the formal statement of the implicit function theorem

IMPLICIT FUNCTION THEOREM: *Let Φ be of class $C^{(q)}$ from an open set $D \subset E^n$ into E^{n-r}, where $q \geq 1$ and $1 \leq r < n$. Let $x_0 \in D$ be such that $\Phi(x_0) = 0$ and $|\tilde{\mathbf{J}}\Phi(x_0)| \neq 0$. Then there exists an open set $U \subset E^n$ containing x_0, an open set $V \subset E^r$ containing $\hat{x}_0$, and $\phi = (\phi_1, \phi_2, \ldots, \phi_{n-r})$ of class $C^{(q)}$ on V such that*

$$|\tilde{\mathbf{J}}\Phi(x)| \neq 0 \;\; \textit{for all} \;\; x \in U$$

and

$$\{x \in U : \Phi(x) = 0\} = \{(\hat{x}, \phi(\hat{x})) : \hat{x} \in V\}.$$

Effectively $(\hat{x}, \phi(\hat{x}))$ provides a local coordinatization at the point x_0 on the solution locus $M = \{x \in U : \Phi(x) = 0)\}$ with respect to E^r and $|\tilde{\mathbf{J}}\Phi(x)| \neq 0$ implies condition (ii) of the definition of an r-manifold made above. A proof of a slightly more general version of the implicit function theorem can be found in [Loomis and Sternberg 1975].

It is important to note that while the condition on the rank of the Jacobian $\mathbf{J}\Phi$ being (n-r) at a point x_0 satisfying equations 1 is a sufficient condition for there to exist a neighborhood about x_0 being an r-manifold, it is not a necessary condition. It is possible for the rank of $\mathbf{J}\Phi$ to be less than (n-r) at x_0 and still have a neighborhood of x_0 being an r-manifold. If the condition on the rank of the Jacobian is violated, further analysis of the higher order derivatives of the functions Φ_i with respect to feature space coordinates is required. This will be reserved for future work and will not be presented here.

Even though the Jacobian for a set of class $C^{(q)}$ equations may not be of sufficient rank for the entire set, the implicit function theorem is still useful for establishing upper bounds on the dimension of measurement ambiguity at a point on the solution locus. In fact if the actual rank of the Jacobian $\mathbf{J}\Phi$ is w in the feature space E^n, a value of an upper bound on the dimension of measurement ambiguity is clearly (n-w). From a large number of successive images it is possible to get a set of equations which number far greater than n but yet have their respective Jacobian with rank $w < n$. This may determine a search for a different set of preselected physical parameters to alter the state of the imaging system so that the resulting equations from successive images have a smaller upper bound on the dimension of measurement ambiguity at certain points. Reduction of even a single dimension of ambiguity results in a far superior generalized stereo method. Clearly a necessary condition for a unique n-tuple feature state point measurement by a given generalized stereo technique is that the dimensionality be zero at that point if in fact the dimension of measurement ambiguity is defined there.

4 MEASUREMENT AMBIGUITY AND SYMMETRY

In section 2 generalized stereo was presented at a level of abstraction which unifies existing conventional stereo techniques such as parallax and photometric stereo and also which serves as a general paradigm for developing a broader variety of stereo techniques. As the key goal of any generalized stereo technique is to optimally disambiguate the physical measurement of a given visual object feature this new general formulation of stereo vision poses some serious mathematical challenges aimed at characterizing the conditions under which disambiguation is feasible. This section takes a formal look at the role that the *symmetry* of solution loci play in characterizing these conditions for measurement disambiguation. Inspired by the *Erlangen programm* of Felix Klein begun by his famous address at Erlangen in 1872, generalized stereo will be formulated at a higher level of abstraction still, using the notion of the *action* of a group of symmetries on the n-tuple point field of feature space.

4.1 Background

The main tool used by Klein's *Erlangen programm*, to describe any type of geometry specified by a set of constraining axioms on a point field, is the group of *automorphisms* (i.e. one-to-one correspondences) of the point field into itself that preserves the validity of all the constraining axioms. It is understood that a point field such as a plane exists independent of the way it is coordinatized. A *coordinatization* of a point field is a one-to-one correspondence between the point field and a set of coordinate symbols

$$p \rightarrow x \;\; or \;\; x = x(p).$$

In the case of n-dimensional Euclidean point space the set of coordinate symbols are the n-tuples $(x_1, x_2, \ldots, x_n)$. Given an automorphism of the point field

$$\sigma: p \rightarrow p' = \sigma p$$

a new *equivalent* coordinatization is produced

$$x'(p) = x(p') = x(\sigma p).$$

Therefore the automorphism σ is described by the transformation S of coordinates from x to x'

$$x' = S(x) \quad \{x = x(p), \;\; x' = x(p')\}\;.$$

The correspondence $\sigma \rightarrow S$ is called a *realization* of an abstract group Γ of automorphisms of a point field if the correspondence satisfies

$$I \rightarrow E, \qquad \sigma\tau \rightarrow ST, \tag{7}$$

where I is the identity automorphism and E is the identity coordinate transformation and T is a coordinate transformation corresponding to the automorphism τ. A correspondence between any two groups satifying the properties shown in 7 is said to be a *homomorphism*. Hence a realization of a group Γ is a homomorphism of Γ into the group of coordinate transformations. In particular, if the realization is defined by an injective (i.e. one-to-one) homomorphism, then the realization is said to be *faithful*. The abstract group Γ can be considered as an entity unto itself. For a good introduction to group theory consult [Jacobson 1974] or [Herstein 1975]. The correspondence between automorphisms and coordinate transformations satisfied by the relations in 7 is also said to define an *action* of the group Γ on the point field.

The abstract automorphism group that leaves the axioms of n dimensional Euclidean geometry invariant, for example, is the *normalizer* subgroup of the group of proper orthogonal linear transformations, $O^{+}(n)$, as a subgroup of the full linear group $GL(n)$ of invertible nxn matrices. The *normalizer* subgroup of a subgroup G' of a group G is defined to be the subset of G

$$\{g \in G : g^{-1}hg \in G' \;\; for\ all\ h \in G'\}\;.$$

It is clear that this subset forms a subgroup of G which at least includes G'. The normalizer subgroup of $O^{+}(n)$ also includes dilations (i.e. uniform change of scale) and improper orthogonal rotations (i.e. 'reflections'). It is intuitively clear that the *congruence* of figures in E^n constructed from n-vectors is preserved under the action of this subgroup of linear transformations. For a more in depth discussion of the group theoretic invariance of Euclidean geometry and other invariants of groups consult [Weyl 1946].

The goal of the *Erlangen programm* was to solve any problem in a particular geometry purely in terms of the group theoretic properties of the transformations (i.e. automorphisms) that preserve its objective nature by leaving its defining axioms invariant. With a similar philosophy in mind the problem of characterizing when an ambiguous solution locus can be disambiguated is formulated purely in terms of the interaction of the group of automorphisms acting on a feature space point field that leaves invariant the ambiguous measurement solution locus for an object feature, with the set of *stereo automorphisms* induced on feature space by image system state transitions. The theoretical treatment is very general but has concrete applications in supplying intuition towards selecting image state transitions that disambiguate well defined symmetrics possessed by a solution locus. Indeed any kind of symmetry inherent to a measurement solution locus is a representation of ambiguity.

The automorphisms leaving invariant an ambiguous solution locus for an object feature from a single image preserves the physical reality of the measurement with respect to the physical state of the imaging system and the evaluation of the image operation functional that is used. For example, consider the equireflectance measurement locus of local surface orientation in a feature space point field coordinatized with gradient space coordinate symbols as in figure 2. With respect to the observed image irradiance and the given state of the imaging system in terms of the imaging geometry etc., any point 2-tuple state (p,q) lying on the equireflectance locus can be replaced by any other point 2-tuple state (p',q') also lying on the locus. Measurement ambiguity implies a

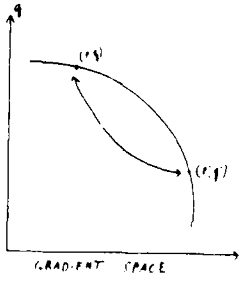


Figure 2

virtual indistinguishability between distinct measurement states on a solution locus which can be represented by the *symmetries* of one-to-one correspondences of the feature space point field into itself leaving the solution locus invariant. Generalized stereo techniques attempt to generate a series of solution loci produced from image state transitions that mutually do not possess the same symmetries. It is clear that for zero measurement error which is assumed here, no matter how many solution loci are generated, any pair of loci will have at least one point in common.

The set of automorphisms leaving a solution locus invariant precisely describe the physical symmetries that are inherent to the measurement of an object feature with respect to a given imaging system model. Figure 1 from [Wolff 1987a] illustrates the measurement of surface orientation using the Torrance-Sparrow reflectance model by varying only the incident linear polarization of the light source. Regardless of the number of successive images taken corresponding to different incident linear polarizations varying between parallel and perpendicular orientation with respect to the plane of incidence, the equireflectance measurement curves will pass through the exact same two measurement point states. The physical reason for this results from the invariance of observed image irradiance when the angle between the local orientation vector and the plane of incidence determined by the light source and viewing vectors, is constant regardless of which side of the plane of incidence the orientation vector lies. This physical symmetry manifests itself as the invariance of the equireflectance solution locus under the automorphism represented as the improper orthogonal rotation

$$\begin{pmatrix} \frac{(p_s)^2-(q_s)^2}{(p_s)^2+(q_s)^2} & \frac{2p_sq_s}{(p_s)^2+(q_s)^2} \\ \frac{2p_sq_s}{(p_s)^2+(q_s)^2} & \frac{(q_s)^2-(p_s)^2}{(p_s)^2+(q_s)^2} \end{pmatrix}$$

in gradient space coordinates. It is quite evident that the variation of the physical parameter representing incident linear polarization in the image irradiance equation does not alter this inherent symmetry. Therefore it is impossible to resolve between two distinct but indistinguishable measurement points that are transformed to each other by 'reflection'.

4.2 The Symmetry Group For A Solution Locus

The definition of a symmetry element for a solution locus M_0 is now formalized. Feature space of n-dimensions will be represented by E^n and unless otherwise specified should be interpreted as only a point field. An automorphism of E^n leaving the subset M_0 invariant is said to *stabilize* M_0. Denote the set of all automorphisms of E^n by A_{E^n} and the set of all automorphisms stabilizing M_0 by A_{M_0}. Clearly A_{M_0} forms a subgroup of A_{E^n}.

Definition 1 *A symmetry element with respect to a nonempty subset M_0 of E^n is an equivalence class of automorphisms that stabilize M_0 defined by the relation that two automorphisms are equivalent if and only if they nave exactly the same action when restricted to M_0.*

The entire set of symmetry elements of M_0, denoted by S_{M_0}, form a group. Letting ϵ_{M_0} denote the equivalence class of automorphisms in A_{M_0} which are the identity action on M_0 it can be easily shown that ϵ_{M_0} forms a *normal* subgroup of A_{M_0} and that

$$S_{M_0} \approx A_{M_0}/\epsilon_{M_0}$$

where the ≈ sign represents isomorphism. That is, S_{M_0} can be realized as the cosets of ϵ_{M_0} in A_{M_0}. The correspondence, ϕ_{can}, defined by mapping a coset realizing an element of S_{M_0} into the automorphism in A_{M_0} with the same action on M_0 and which is the identity action on the complement of M_0 in E^n, is an injective homomorphism of S_{M_0} into A_{M_0} thus making S_{M_0} a subgroup of A_{M_0}. [5] Since to each point in the feature space point field there is assigned a unique coordinate symbol, the injective homomorphism ϕ_{can} induces a faithful realization of S_{M_0} in terms of a set of coordinate transformations on feature space. The realization ϕ_{can} will be referred to as the *canonical realization* of S_{M_0} in A_{E^n}.

A natural generalization to the definition of symmetry elements which are mutual to a finite collection of nonempty pairwise intersecting subsets of E^n is as follows

Definition 2 *A mutual symmetry element with respect to nonempty subsets $M_0, M_1, \ldots, M_l$ of E^n, where $M_i \cap M_j \neq \emptyset$ for all $i < j$, is an equivalence class of automorphisms of E^n stabilizing each $M_0, M_1, \ldots, M_l$ individually, defined by the relation that two automorphisms are equivalent if and only if the action of these two automorphisms is exactly the same on $\cap_{i=0}^{l} M_i$.*

Denote the set of automorphisms of E^n stabilizing $M_0, M_1, \ldots, M_l$ individually by $A_{M_0,M_1,\ldots,M_l}$. It is straightforward to demonstrate that $A_{M_0,M_1,\ldots,M_l}$ forms a subgroup of A_{E^n}. The following lemma establishes the group theoretic equivalence between the set of mutual symmetry elements with respect to $M_0, M_1, \ldots, M_l$ and the symmetry group $S_{\cap_{i=0}^{l} M_i}$ up to isomorphism.

Lemma 1 *Let P be the set of mutual symmetry elements with respect to nonempty pairwise intersecting subsets $M_0, M_1, \ldots, M_l$ of E^n. Then P forms a group isomorphic to $S_{\cap_{i=0}^{l} M_i}$.*

Proof: It is first demonstrated that any automorphism of $A_{M_0,M_1,\ldots,M_l}$ must in turn stabilize $\cap_{i=0}^{l} M_i$ and therefore be a member of $A_{\cap_{i=0}^{l} M_i}$. It is then demonstrated that the equivalence classes of mutual symmetry elements of $A_{M_0,M_1,\ldots,M_l}$ are simply restrictions of the equivalence classes of symmetry elements of $A_{\cap_{i=0}^{l} M_i}$ to automorphisms with the additional constraint that they stabilize $M_0, M_1, \ldots, M_l$ individually. This restriction mapping provides an isomorphism.

Let p_a be an automorphism in $A_{M_0,M_1,\ldots,M_l}$ and let $p_a(M_i)$ denote the range of the action of p_a on the subset M_i. Thus $p_a(M_i) = M_i$ for all $1 \leq i \leq l$. Since $p_a(\cap_{i=0}^{l} M_i) \subset M_i$ for all $1 \leq i \leq l$, clearly $p_a(\cap_{i=0}^{l} M_i) \subset \cap_{i=0}^{l} M_i$. To show that $p_a(\cap_{i=0}^{l} M_i) = \cap_{i=0}^{l} M_i$, suppose there exists a point $x \notin \cap_{i=0}^{l} M_i$ such that $p_a(x) \in \cap_{i=0}^{l} M_i$. Since $A_{M_0,M_1,\ldots,M_l}$ is a subgroup, $p_a^{-1}(\cap_{i=0}^{l} M_i) \subset \cap_{i=0}^{l} M_i$ which contradicts the fact that $p_a^{-1}(p_a(x)) \notin \cap_{i=0}^{l} M_i$. Therefore, since p_a is a one-to-one correspondence, $p_a(\cap_{i=0}^{l} M_i) = \cap_{i=0}^{l} M_i$.

Since each element of $A_{M_0,M_1,\ldots,M_l}$ stabilizes $\cap_{i=0}^{l} M_i$ it is clear that a mutual symmetry element equivalence class with respect to $M_0, M_1, \ldots, M_l$ is a restriction of the equivalence class of a symmetry element of $S_{\cap_{i=0}{}^{l} M_i}$ having the same exact action on $\cap_{i=0}^{l} M_i$ being further constrained by having to stabilize $M_0, M_1, \ldots, M_l$ individually. It is clear that this restriction mapping is one-to-one, onto. The fact that $A_{M_0,M_1,\ldots,M_l}$ is a subgroup and therefore has the closure property implies that the restriction mapping is a homomorphism.

Q.E.D.

[5] In general group G' is a subgroup of group G if there exists an injective homomorphism of G' into G.

4.3 Stereo Automorphisms And The Preserved Symmetry Subgroup

Stereo vision is a dynamic process in the sense that the physical state of the imaging system is always altered between successive images. Section 2 showed this as the transition from one m-tuple state to another m-tuple state

$$(a_1, a_2, \ldots, a_m) \rightarrow (a'_1, a'_2, \ldots, a'_m)\ .$$

An equivalent way of representing this state transition with respect to its effect on the measurement solution locus is in terms of a corresponding one-to-one, onto coordinate transformation in feature space

$$\begin{array}{l} x_1 \rightarrow x'_1(x_1, x_2, \ldots, x_n) \\ x_2 \rightarrow x'_2(x_1, x_2, \ldots, x_n) \\ \qquad \vdots \\ x_n \rightarrow x'_n(x_1, x_2, \ldots, x_n) \end{array} \qquad (8)$$

where the coordinate transformation functions, x'_i, satisfy the constraint that $(x_1, x_2, \ldots, x_n)$ solves the system

$$\left.\begin{array}{c} \Phi_1(x_1, x_2, \ldots, x_n, a'_1, a'_2, \ldots, a'_m, F'_t(I_s)|_A) = 0 \\ \Phi_2(x_1, x_2, \ldots, x_n, a'_1, a'_2, \ldots, a'_m, F'_t(I_s)|_A) = 0 \\ \vdots \\ \Phi_h(x_1, x_2, \ldots, x_n, a'_1, a'_2, \ldots, a'_m, F'_t(I_s)|_A) = 0 \end{array}\right\}$$

if and only if it solves the system (9).

These systems of equations correspond to the same equations 1 in section 2 except without the superscript (k) for ease of reading. The coordinate transformation 8 therefore describes the transformation of the measurement solution locus M when the imaging system is in state $(a_1, a_2, \ldots, a_m)$ to the measurement solution locus M' when the imaging system state is changed to $(a'_1, a'_2, \ldots, a'_m)$. While many coordinate transformations depicted in 8 may satisfy the above constraint all such transformations are equivalent in the sense that the range of any coordinate transformation when restricted to M is M'. This motivates the following definition

Definition 3 *Given a generalized stereo technique let M and M' be two solution loci for an object feature state in E^n with respect to two physical states of the imaging system. Then the set of automorphisms of A_{E^n} whose range when restricted to M is M' are termed stereo automorphisms. Two stereo automorphisms are equivalent if when restricted to a solution locus M they have exactly the same range.*

For a given generalized stereo technique the set of all stereo automorphisms are generated by all possible physical state transitions of the imaging system. The set of all stereo automorphisms does not necessarily form a group. The only group property which this set may not satisfy is closure. A stereo automorphism bringing solution locus M_i into solution locus M_j will be denoted by t_{ij} and its action is shown as $t_{ij}(M_i) = M_j$. The process of disambiguation for a generalized stereo vision technique can be characterized by the algebraic interaction of the set of stereo automorphisms with the symmetry group S_M for the measurement solution locus M obtained from any successive image.

Recall from section 2 that the solution locus M_0 is defined to be *disambiguated* by $l+1$ successive images (i.e. l image state transitions) if and only if

$$\cap_{i=0}^{l} M_i \subsetneq M_0\ .$$

In the same vein, the solution locus M_0 is equivalently defined to be disambiguated by stereo automorphisms $\{t_{01}, t_{02}, \ldots, t_{0l}\}$. Before making another key definition an important lemma is proved.

Lemma 2 *There exists an injective homomorphism of $S_{\cap_{i=0}^{l} M_i}$ into S_{M_0} thus making $S_{\cap_{i=0}^{l} M_i}$ a subgroup of S_{M_0}. The solution locus M_0 is disambiguated by stereo automorphisms $\{t_{01}, t_{02}, \ldots, t_{0l}\}$ if and only if $S_{\cap_{i=0}^{l} M_i}$ is a proper subgroup of S_{M_0}.*

Proof: Construct an injective homomorphism using the coset realization of $S_{\cap_{i=0}^{l} M_i}$ and S_{M_0} using isomorphisms ϕ^1_{iso} and ϕ^2_{iso} respectively. Define the map ϕ_r by assigning a coset of feature space automorphisms representing an element of $S_{\cap_{i=0}^{l} M_i}$ with the element of S_{M_0} represented by the coset of feature space automorphisms having the equivalent action on the subset $\cap_{i=0}^{l} M_i$ and being the identity mapping on the difference subset $M_0 - \cap_{i=0}^{l} M_i$. It is clear that this mapping is an injective homomorphism. The injective homomorphism ϕ_s results from the following commutative diagram

$$\begin{tikzcd}
A_{\cap_{i=0}^{l} M_i}/e_{\cap_{i=0}^{l} M_i} \arrow[r, "\phi_r"'] & A_{M_0}/e_{M_0} \arrow[d, "(\phi^2_{iso})^{-1}"] \\
S_{\cap_{i=0}^{l} M_i} \arrow[u, "\phi^1_{iso}"] \arrow[r, "\phi_s"] & S_{M_0}
\end{tikzcd}$$

The range of this injective homomorphism is a proper subgroup of S_{M_0} if and only if $M_0 - \cap_{i=0}^{l} M_i$ is non-empty or equivalently $\cap_{i=0}^{l} M_i \subset M_0$.

Q.E.D.

Lemmata 1 and 2 together imply the following commutative diagram

$$\begin{tikzcd}
S_{\cap_{i=0}^{l} M_i} \arrow[r, "\phi_s"] & S_{M_0} \\
P \arrow[u, "\phi_{iso}"] \arrow[ur, "\phi_p"'] &
\end{tikzcd}$$

Injective homomorphism ϕ_p is defined as the composition of isomorphism ϕ_{iso}, which is the restriction mapping from lemma 1, with injective homomorphism ϕ_s, defined in lemma 2.

$$\left.\begin{array}{c} \Phi_1(x'_1(x_1, x_2, \ldots, x_n), x'_2(x_1, x_2, \ldots, x_n), \ldots, x'_n(x_1, x_2, \ldots, x_n), a_1, a_2, \ldots, a_m, F_t(I_s)|_A) = 0 \\ \Phi_2(x'_1(x_1, x_2, \ldots, x_n), x'_2(x_1, x_2, \ldots, x_n), \ldots, x'_n(x_1, x_2, \ldots, x_n), a_1, a_2, \ldots, a_m, F_t(I_s)|_A) = 0 \\ \vdots \\ \Phi_h(x'_1(x_1, x_2, \ldots, x_n), x'_2(x_1, x_2, \ldots, x_n), \ldots, x'_n(x_1, x_2, \ldots, x_n), a_1, a_2, \ldots, a_m, F_t(I_s)|_A) = 0 \end{array}\right\} (9)$$

Definition 4 *The set of* *preserved symmetry elements* *of the symm group S_{M_0} for measurement solution locus M_0 with respect to stereo automorphisms $\{t_{01}, t_{02}, \ldots, t_{0l}\}$ is the range of the injective homomorphism ϕ_p from P into S_{M_0} defined in the commutative diagram immediately above.*

Clearly the set of preserved symmetry elements defined above form a subgroup of S_{M_0} isomorphic to $S_{\cap_{i=0}^{l} M_i}$. The canonical realization of the set of preserved symmetry elements with respect to stereo automorphisms $\{t_{01}, t_{02}, \ldots, t_{0l}\}$ in A_{E^n} is the restriction of ϕ_{can} to the homomorphic image of P in S_{M_0} with respect to ϕ_p.

4.4 Disambiguation Theorems

The tools have been developed to help prove some precise statements about the conditions under which measurement disambiguation takes place.

Theorem 1 *An element s of the symmetry group S_{M_0} for solution locus M_0 with canonical realization p is a preserved symmetry element with respect to stereo automorphisms $\{t_{01}, t_{02}, \ldots, t_{0l}\}$ if and only if $t_{0i}^{-1} p t_{0i}$ is a canonical realization for some element of S_{M_0} for all $1 \leq i \leq l$.*

Proof: Suppose that s is a preserved symmetry element of symmetry group S_{M_0} with canonical realization p. From the definition of ϕ_r in lemma 2 $p \in A_{E^n}$ is a canonical realization of a preserved symmetry element if and only if $p(\cap_{i=0}^{l} M_i) = \cap_{i=0}^{l} M_i$ and p is the identity action on the complement of $\cap_{i=0}^{l} M_i$ in E^n. Hence, $p(M_i) = p(t_{0i}(M_0)) = t_{0i}(M_0)$ for all $1 \leq i \leq l$. Therefore, $t_{0i}^{-1}(p(t_{0i}(M_0))) = t_{0i}^{-1}(t_{0i}(M_0))$ and from the definition of the action of a group on a set $(t_{0i}^{-1} p t_{0i})(M_0) = M_0$. That is, $t_{0i}^{-1} p t_{0i}$ stabilizes M_0. It needs to be shown that $t_{0i}^{-1} p t_{0i}$ is the identity action on the complement of M_0 in E^n. The fact that p is the identity action on the complement of $\cap_{i=0}^{l} M_i$ and that for $x \notin M_0$, $t_{0i}(x) \notin M_i$, means that $(t_{0i}^{-1} p t_{0i})(x) = t_{0i}^{-1}(t_{0i}(x)) = x$.

Suppose that $t_{0i}^{-1} p t_{0i}$ is a canonical representation for some element of S_{M_0} for all $1 \leq i \leq l$. Then $(t_{0i}^{-1} p t_{0i})(M_0) = M_0$ and by doing the reverse algebra of the previous paragraph $p(M_i) = M_i$ for all $1 \leq i \leq l$. From the argument of lemma 1, this implies that $p(\cap_{i=0}^{l} M_i) = \cap_{i=0}^{l} M_i$. Now suppose that p does not have the identity action on the complement of $\cap_{i=0}^{l} M_i$. Since p is already a canonical realization for an element of S_{M_0}, it must at least have the identity action on the complement of M_0. Therefore there must exist two distinct points $x, \tilde{x} \in M_0$, $x, \tilde{x} \notin \cap_{i=0}^{l} M_i$ such that $p(x) = \tilde{x}$. This means there exists a $1 \leq k \leq l$ such that $x \notin M_k$. Hence $t_{0k}^{-1}(x) \notin M_0$ because the range of t_{0k} acting on M_0 is M_k. Since t_{0k} is a bijection, $t_{0k}^{-1}(x)$ and $t_{0k}^{-1}(\tilde{x})$ must be distinct. Since $(t_{0k}^{-1} p t_{0k}) t_{0k}^{-1}(x) = t_{0k}^{-1}(\tilde{x})$, $t_{0k}^{-1} p t_{0k}$ could not possibly be a canonical realization of an element of S_{M_0} since it is not the identity action on $t_{0k}^{-1}(x) \notin M_0$. Contradiction. Hence p must be the identity action on the complement of $\cap_{i=0}^{l} M_i$ and therefore a canonical realization of a preserved symmetry element.

Q.E.D.

The following theorem establishes a purely algebraic relationship between the set of stereo automorphisms and the symmetry group S_{M_0} that characterizes when measurement disambiguation takes place.

Theorem 2 *For a given generalized stereo technique a measurement solution locus M_0 is disambiguated by stereo automorphisms $\{t_{01}, t_{02}, \ldots, t_{0l}\}$ corresponding to l state transitions of preselected imaging parameters if and only if there exists a stereo automorphism t_{0i}, $1 \leq i \leq l$, such that t_{0i} is* **not** *in the normalizer subgroup of the canonical realization of S_{M_0} in A_{E^n}.*

Proof: Lemma 2 shows that a solution locus M_0 is disambiguated by stereo automorphisms $\{t_{01}, t_{02}, \ldots, t_{0l}\}$ if and only if $S_{\cap_{i=0}^{l} M_i}$ is a proper subgroup of S_{M_0}. Lemma 1 shows that the set of mutual symmetry elements, P, with respect to M_0 and $M_i = t_{0i} M_0$, $1 \leq i \leq l$ is isomorphic to $S_{\cap_{i=0}^{l} M_i}$. Combining these results with the above commutative diagrams implies that a solution locus M_0 is disambiguated by stereo automorphisms $\{t_{01}, t_{02}, \ldots, t_{0l}\}$ if and only if the set of canonical realizations for preserved symmetry elements is a proper subgroup of the group of canonical realizations for the symmetry group S_{M_0} in A_{E^n}.

Suppose that M_0 is not disambiguated by stereo automorphisms $\{t_{01}, t_{02}, \ldots, t_{0l}\}$ meaning the set of canonical realizations for preserved symmetry elements is equal to the set of canonical realizations of the group S_{M_0} in A_{E^n}. From theorem 1, this implies that each t_{0i}, $1 \leq i \leq l$, is in the normalizer subgroup of the group of canonical realizations for S_{M_0} in A_{E^n}. Stating the contrapositive, if there exists a stereo automorphism t_{0i} **not** in the normalizer subgroup of the group of canonical realizations for S_{M_0} in A_{E^n} then M_0 is disambiguated by stereo automorphisms $\{t_{01}, t_{02}, \ldots, t_{0l}\}$.

The proof of the other direction is similar. Suppose that each stereo automorphism t_{0i}, $1 \leq i \leq l$, is in the normalizer subgroup of the canonical realizations of S_{M_0} in A_{E^n}. Then from theorem 1 each canonical realization corresponding to an element in S_{M_0} is a canonical realization for a preserved element implying that solution locus M_0 cannot be disambiguated. The contrapositive of this statement proves the rest of the theorem.

Q.E.D.

5 CONCLUSION

A generalized axiomatic definition of stereo vision was presented that was shown not only to unify existing stereo techniques but to lead to a multitude of new stereo techniques not yet considered. Several new examples of this generalized variety of stereo vision were given. A given generalized stereo technique is completely determined by (i) a *visual object feature* to be measured, (ii) an *image operation functional* (iii) a preselected set of physically independent *imaging parameters* at least one of which is to be varied between each successive image and (iv) a system of *camera modeling equations*. Previously only geometric imaging parameters were varied in stereo techniques to measure object features such as position or local surface orientation. Stereo techniques that measured surface orientation were presented that varied physical imaging parameters such as incident wavelength and/or polarization and it was suggested that surface orientation could also be measured by varying object surface temperature or external/internal magnetic and/or electric fields. The direct measurement of surface curvature at a point on a surface, independent of knowledge about neighboring surface normals, could also be accomplished from the variation of any of these imaging parameters using a 3-tuple image operation functional consisting of image irradiance and the two components of the image gradient. This by no means saturates all the possibilities for generalized stereo techniques.

The central theme of measurement of object features by a generalized stereo technique is disambiguation of measurement solution loci in feature space. Theoretical developments were presented to characterize the "size" of measurement ambiguity at a point on a solution locus as well as the conditions under which measurement disambiguation takes place. With a complete norm defined on feature space, measurement solution loci were considered as the union of embedded sub-manifolds and one way of characterizing the "size" of measurement ambiguity at a point is the evaluation of the dimension of a sub-manifold neighborhood containing that point, if in fact one existed. The implicit function theorem was shown to provide an algorithm for evaluating an upper bound on the dimension of measurement ambiguity at a point.

In section 4 the formalism for generalized stereo was recast into the language of group theory emulating the elegant philosophy of Klein's *Erlangen programm*. Measurement ambiguity of a solution locus was equivalently presented as the manifestation of symmetry which is described by its corresponding symmetry group and represented in terms of feature space automorphisms that leave the solution locus invariant. Imaging system state transitions take the form of feature space automorphisms which attempt to break up symmetries of solution loci thus reducing measurement ambiguity. When expressed in this formalism very precise statements can be made about the conditions under which a given generalized stereo technique can disambiguate a measurement solution locus. A key theorem was proven using the canonical realization of the symmetry group in terms of feature space automorphisms for a solution locus stating that an image state transition disambiguates a solution locus if and only if its corresponding stereo automorphism is not in the normalizer subgroup of the realization of the symmetry group for the solution locus.

This paper is merely an introduction to the application and theory of generalized stereo techniques. Much work remains to discover and implement new stereo techniques as well as to develop a rich mathematical formalism that embodies these techniques.

ACKNOWLEDGEMENTS

I would like to extend my appreciation to Kicha Ganapathy for letting me pursue many portions of the work contained in this paper while at AT&T Bell Laboratories. I would like to thank Terry Boult and John Kender for their continual support and encouragement. I have also benefitted from discussions with Paul Hriljac and Maurice Hasson.

References

[Aloimonos and Swain 1985] Aloimonos, J., and Swain, M., *Shape From Texture*, IJCAI 1985, pp.926-931.

[Ballard and Brown 1982] Ballard, D.H., and Brown, C.M., *Computer Vision*, Prentice-Hall, 1982.

[Do Carmo 1976] Do Carmo, M.P., *Differential Geometry of Curves and Surfaces*, Prentice-Hall, 1976.

[Herstein 1975] Herstein, I.N., *Topics in Algebra*, Xerox Press, 1975.

[Ikeuchi 1987] Ikeuchi, K., *Precompiling a Geometrical Model into an Interpretation Tree for Object Recognition in Bin-Picking Tasks*, DARPA IUW, pp.321-339, 1987.

[Jacobson 1974] Jacobson, N., *Basic Algebra I*, Freeman Press, 1974.

[Kender 1980] Kender, J.R., *Shape From Texture*, PhD Thesis, CMU, November 1980.

[Loomis and Sternberg 1975] Loomis L.M., Sternberg, S., *Advanced Calculus*, Addison-Weseley, 1975

[Ohta et al. 1980] Ohta, Y.I., et al., *Obtaining Surface Orientation from Texels Under Perspective Projection*, IJCAI, 1980, pp. 746-751.

[Schwartz 1983] Schwartz, J.T., *Structured Light Sensors For 3-D Robot Vision*, NYU Tech Report no. 65, March 1983.

[Silver 1980] Silver, W.M., *Determining Shape and Reflectance Using Multiple Images*, M.sc dissertation, AI Lab, M.I.T., 1980.

[Strat and Fischler 1985] Strat, T.M., *One-Eyed Stereo: A General Approach To Modeling 3-D Scene Geometry*, DARPA IUW 1985, pp.363-372.

[Torrance and Sparrow 1967] Torrance, K.E., and Sparrow, E.M., *Theory for Off-Specular Reflection From Roughened Surfaces*, Journal of The Optical Society of America, vol.57 #9 pp.1105-1114, September 1967.

[Weyl 1946] Weyl, H., *The Classical Groups, Their Representations and Invariants*, Princeton University Press, 1946.

[Wolff 1986] Wolff, L.B., *Physical Stereo For Combined Specular and Diffuse Reflection*, Columbia Univ. Tech Report CS-242-86, Nov. 1, 1986.

[Wolff 1987a] Wolff, L.B., *Spectral and Polarization Stereo Methods Using A Single Light Source*, ICCV 1987, pp. 708-715.

[Wolff 1987b] Wolff, L.B., *Surface Curvature and Contour From Photometric Stereo*, DARPA image understanding workshop, Feb. 1987, pp.821-824.

[Wolff 1987c] Wolff, L.B., *Surface Orientation From Polarization Images*, SPIE 1987, Optics, Illumination, and Image Sensing for Machine Vision II.

[Wolff 1988] Wolff, L.B., *Accurate Measurement of Second Order Variations in Surfaces Using Reflectance Maps*, Under Preparation.

[Woodham 1978] Woodham, R.J.,*Reflectance Map Techniques For Analyzing Surface Defects in Metal Castings*, M.I.T. AI-TR-457, June 1978.

STOCHASTIC STEREO MATCHING OVER SCALE

Stephen T. Barnard

Artificial Intelligence Center
SRI International
Menlo Park, California 94025

415-859-4720
barnard@sri

February 4, 1988

Abstract

A stochastic optimization approach to stereo matching is presented. Unlike conventional correlation matching and feature matching, the approach provides a dense array of disparities, eliminating the need for interpolation. First, the stereo matching problem is defined in terms of finding a disparity map that satisfies two competing constraints: (1) matched points should have similar image intensity, and (2) the disparity map should vary as slowly as possible. These constraints are interpreted as specifying the potential energy of a system of oscillators. Ground states are approximated by a new variant of simulated annealing, which has two important features. First, the microcanonical ensemble is simulated using a new algorithm that is more efficient and more easily implemented than the familiar Metropolis algorithm (which simulates the canonical ensemble). Secondly, it uses a hierarchical, coarse-to-fine control structure employing laplacian pyramids of the stereo images. In this way, quickly computed results at low resolutions are used to initialize the system at higher resolutions.

1 Introduction

Few problems in computational vision have been investigated more vigorously than stereo. Compared to other modes of depth perception, stereo vision seems relatively straightforward. The images received by two eyes are slightly different due to binocular parallax; that is, they exhibit a disparity that varies over the visual field, and that is inversely related to the distance of imaged points from the observer. If we can determine this disparity field we can measure depth and mimic human stereo vision.

This paper describes an approach to stereo in which the matching problem is posed as computational analogy to a thermodynamic physical system. The state of the system encodes a disparity map that specifies the correspondence between the images. Each such state has an energy that provides a heuristic measure of the "quality" of the correspondence. To solve the stereo matching problem, one looks for the ground state; that is, the state (or states) of lowest potential energy.

The remainder of this section briefly discusses the major approaches to stereo matching. In Section 2 the model system for the stochastic optimization method is defined. Section 3 describes how a stochastic technique called simulated annealing can be used to perform the optimization and introduces a new, more efficient variety of simulated annealing, called microcanonical, hierarchical annealing. This method samples a microcanonical ensemble over a sequence of increasingly finer scales. Several experimental results are given in Section 4. Section 5 concludes with some observations.

1.1 Background

The development of computational models of stereo vision has been guided by both scientific and technological motivations. The modularity of stereopsis in the human visual system, conclusively demonstrated by random-dot stereograms, indicates that this perceptual function can be studied in isolation. If the same computational principles used for stereo also apply to other modes of perception a successful model of stereo could suggest models for other vision problems. Stereo finds important practical applications in mapping and robot sensing.

1.2 Correlation

Perhaps the most obvious approach to stereo matching, loosely called "correlation," to choose intensity patches in one image and then to search for the best matching location in the other image, typically using normalized cross-correlation as a measure of similarity or mean-square-difference as a measure of dissimilarity. Many variations of this basic theme have been explored.

This general approach suffers from some difficult problems.

1. The size of the patches affects the likelihood of false matches. A patch must be large enough to contain the information necessary to specify another patch unambiguously; or, failing this, some additional means of disambiguating false matches must be used.

2. At the same time, the patches must be small compared to the variation in the disparity map. If the patches are too large the system will be insensitive to significant relief in the scene. These problems have motivated the use of scale hierarchies. (See 1.4 below.)

Support for this work was provided by the Defense Advanced Research Projects Agency under contracts DCA76-85-C-0004 and MDA903-86-C-0084.

3. In typical images much of the area consists of uniform or slowly varying intensity, and correlation will not be sensitive in such cases. In practice, a correlation method can provide only a relatively sparse set of correspondences, from which a dense map must then be interpolated.

1.3 Feature Matching

Another approach is to attempt matching only on "information-rich" points. Even in correlation methods an interest operator is often used to screen patches. The feature-matching approach seeks to establish correspondences directly between discrete sets of points — typicaliy, the output of an edge detector, such as zero-crossing contours.

This approach suffers from similar difficulties:

1. The support of the feature detector affects the likelihood of false matches. Zero-crossings from high-frequency bands will probably have many ambiguous matches for significant ranges of disparity.

2. The support of the feature detector must be small compared to the variation in the disparity map (caused by relief in the scene) if the 2D features are to locate 3D features accurately.

3. Feature matching provides sparse matches by definition.

1.4 Scale Hierarchy

Disparity scales linearly. This suggests that a stereo matcher can begin its search at a coarse scale, find coarsely-quantized disparities, use this result to initialize its search at a finer scale, and so on. In addition to improving efficiency by limiting the effective search space of the matcher, this technique ameliorates the false target problem. Coarse-to-fine control strategies have been used in both correlation and feature-matching models.

1.5 Lattice and Variational Models

Several models of stereo vision fit neither the correlation nor the feature-matching paradigms; instead, they pose the matching problem in terms of optimizing a global measure [1,2,3]. To take one example, Julesz proposed a model consisting of two lattices of spring-loaded magnetic dipoles, representing the two images of a random-dot stereogram [2]. The polarity of the dipoles represents whether pixels in the left and right images are black or white. A state of global fusion is achieved in the ground state, with the attraction or repulsion of the dipoles balanced by the forces of the springs.

More recently, Poggio *et. al.* have proposed a regularization criterion based on minimizing the following quantity [4]:

$$\mathcal{E} = \int\int \{[\nabla^2 G \circ (I_L(x,y) - I_R(x + D(x,y), y))]^2 + \lambda(\nabla D)^2\} dx dy , \tag{1}$$

where I_L and I_R are continuous intensity functions in the left and right visual fields, $\nabla^2 G$ is a linear bandpass filter (laplacian of a gaussian), ∇D is the gradient of disparity, and λ is a constant. Equation (1) can be justified in terms of two heuristics: the first term in the integrand is a measure of photometric difference; the second is a measure of the first-order variation in the disparity map. In this heuristic sense it is similar to the Julesz spring-dipole model, with the two terms corresponding to the potential energy of the dipoles and the springs, respectively. Of course, (1) has the advantage of being precise, as well as addressing the case of continuous intensity.

Witkin *et. al.* described a method for optimizing (1) that is essentially a sophisticated form of gradient descent which tracks the solution over increasingly finer scales [5]. The hope is that $\mathcal{E}$ is convex at a coarse scale and that relatively coarse intermediate solutions will place the system in the correct convex region at finer scales. They report that the method is prone to error when it encounters bifurcations in its trajectory. As the scale becomes finer the system must "choose" which path to follow, and it cannot recover from a mistake because $\mathcal{E}$ may never increase. The solution is therefore critically dependent on initial conditions. This paper presents an alternative stochastic method that can cope with this problem.

2 The Model System

2.1 Epipolar Camera Model

We assume that two coplanar images $\mathcal{L}(x,y)$ and $\mathcal{R}(x,y)$ are formed by central projection with focal length f, and with the centers of projection separated by distance B along a baseline parallel to the image planes. For convenience, we assume that $0 \le x, y < 1$. If a point (x, y, f) in the left image matches point (x', y', f) in the right image (that is, it has disparity $d = x' - x$), the 3D coordinates of the imaged point with respect to the left camera are

$$\mathbf{p} = (\frac{B}{d}x, \frac{B}{d}y, \frac{B}{d}f) .$$

Under these conditions, the disparities are restricted to the horizontal (x) direction. This assumption involves no loss of generality, because if the relative positions and orientations of the two cameras are known, as well as the internal camera parameters, correspondences are restricted to to epipolar lines. If the epipolar lines are not horizontal the images can be mapped into a "normal" stereo pair in which they are.

2.2 Cyclopean Disparity Map

At this point we identify $\mathcal{L}$ and $\mathcal{R}$ with $\nabla^2 G \circ I_L(x,y)$ and $\nabla^2 G \circ I_R(x,y)$ in (1). We seek a disparity map, $\mathcal{D}(x,y)$, defined over the same interval as $\mathcal{L}$ and $\mathcal{R}$, which specifies the correspondence between $\mathcal{L}$ and $\mathcal{R}$. The cyclopean representation introduced by Horn [6] defines $\mathcal{D}$ with the following relation:

$$\mathcal{L}(x - \frac{\mathcal{D}(x,y)}{2}, y) \text{ corresponds to } \mathcal{R}(x + \frac{\mathcal{D}(x,y)}{2}, y) .$$

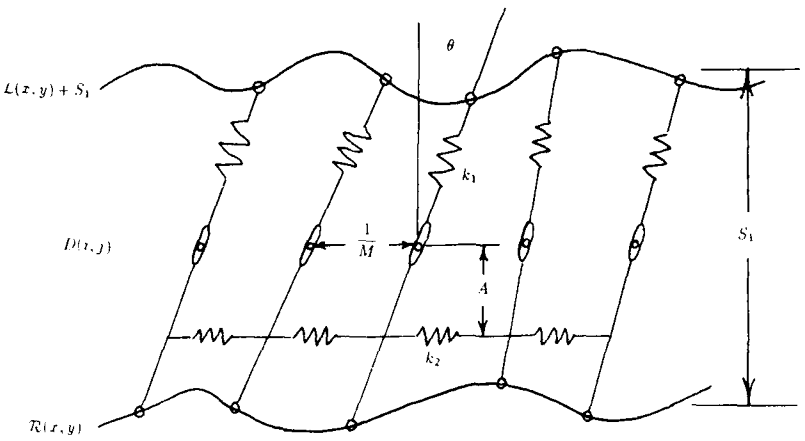
The major advantage of the cyclopean representation is that, by defining disparity without preference to either image, it allows a more uniform treatment of occlusion boundaries.

Rewriting the integrand of equation (1) in the cyclopean form we have:

$$\begin{aligned} \mathcal{E}(x,y) &= [\mathcal{L}(x - \frac{\mathcal{D}(x,y)}{2}, y) - \mathcal{R}(x + \frac{\mathcal{D}(x,y)}{2}, y)]^2 \\ &\quad + \lambda[\nabla\mathcal{D}(x,y)]^2 &(2) \\ &= \mathcal{E}_1(x,y) + \lambda\mathcal{E}_2(x,y) . &(3) \end{aligned}$$

and the quantity to be minimized is

$$\mathcal{E}(\mathcal{D}) = \int\int \mathcal{E}(x,y) dx dy . \tag{4}$$



vertical springs: spring constant k_1, rest length S_1.
horizontal springs: spring constant k_2, rest length $S_2 = \frac{1}{M}$.

Figure 1: Spring model.

2.3 A Spring Model

To establish a concrete idea of the meaning of (3) and (4), consider the spring model illustrated in one dimension in Figure 1.

The model consists of two surfaces, $\mathcal{R}(x,y)$ below and $\mathcal{L}(x,y)+S_1$ above. Midway between these surfaces is a lattice of pivot points, and at each such point is an elastic lever arm, with rest length S_1 and spring constant k_1. The lever arms are free to rotate in the (x,z) plane (i.e., in epipolar planes), and their endpoints are constrained to lie on the two surfaces. The lever arms are connected to their neighbors by other springs with spring constant k_2 which exert a torque over moment arm A. The angles of the lever arms represent disparity on an M^2 cyclopean lattice:

$$D(i,j) = \mathcal{D}(x_i, y_j)\ ,\quad 0 \leq i,j < M\ .$$

with

$$D(i,j) \approx S_1 \sin\theta_{i,j} \approx S_1\theta_{i,j}\ .$$

The potential energy stored in a lever arm is approximately[1]

$$\mathcal{H}_1(i,j) \approx \frac{1}{2}k_1[\mathcal{L}(x - \frac{D(i,j)}{2}, y) - \mathcal{R}(x + \frac{D(i,j)}{2}, y)]^2$$

and in a connecting spring is approximately

$$\mathcal{H}_2(i,j,k,l) \approx \frac{1}{2}k_2\left(\frac{A}{S_1}\right)^2[D(i,j) - D(k,l)]^2\ .$$

The energy associated with a single lattice point is

$$\mathcal{H}(i,j) \approx \mathcal{H}_1(i,j) + \frac{1}{2}\sum_{(k,l)\in\mathcal{N}_{i,j}} \mathcal{H}_2(i,j,k,l)\ . \qquad (5)$$

where $\mathcal{N}_{i,j}$ is the set of neighbors of (i,j). The energy of the entire system is

$$\mathcal{H}(D) = \sum_{i,j}\mathcal{H}(i,j)$$

Comparing terms between (3) and (5), we have approximately

$$\mathcal{H}(D) \propto \mathcal{E}(\mathcal{D})$$

with

$$\lambda = \frac{1}{2}\left(\frac{k_2}{k_1}\right)\left(\frac{A}{S_1}\right)^2$$

We can therefore interpret $\mathcal{E}$ as a hamiltonian specifying the energy of the spring system, neglecting kinetic energy terms. The constant λ is proportional to the relative stiffness of the two types of springs.

A physical realization of the spring model would be a dynamic system of oscillators that would follow a trajectory through a $2M$ dimensional phase space. (Each lever arm has two degrees of freedom: θ and $\dot{\theta}$.) We could flesh out this model by specifying the moments of inertia and damping coefficients of the lever arms. We could also add a periodic forcing function to add energy to the system, balancing the energy dissipated by damping. Having done this, we could could write a hamiltonian, containing both potential terms depending on θ and kinetic terms depending on $\dot{\theta}$, describing the model's deterministic dynamic behavior. In principle, we could trace the trajectory of the system through phase space, gradually reducing the amplitude of the forcing function while keeping the system in dynamic equilibrium. There is little point in simulating the dynamics in such detail, however, since we know that even low-dimensional forced oscillators have chaotic attractors [7]. The dynamics will be effectively stochastic.

An alternative and much less expensive approach, which we shall take in the next two sections, is to explicitly acknowledge

[1] These formulae require a small angle approximation $\sin(\theta) \approx \theta$. Note that the angles can be made arbitrarily small by increasing S_1

the stochastic, ergodic nature of the model. Kinetic energy will be modeled as heat.

3 Stochastic Optimization

We have already partially discretized (1) by defining the lattice D on $\mathcal{D}$. At this point we similarly define lattices L and R on $\mathcal{L}$ and $\mathcal{R}$. D now has integer values and is interpreted as:

$$L(i - \left\lfloor \frac{D(i,j)}{2} \right\rfloor, j) \text{ corresponds to } R(i + \left\lceil \frac{D(i,j)}{2} \right\rceil, j) .$$

Equation (3) becomes

$$E(i,j) = [L(i - \left\lfloor \frac{D(i,j)}{2} \right\rfloor, j) - R(i + \left\lceil \frac{D(i,j)}{2} \right\rceil, j)]^2 + \lambda[\nabla D(i,j)]^2. \tag{6}$$

with

$$[\nabla D(i,j)]^2 = \sum_{k,l \in \mathcal{N}_{i,j}} [D(i,j) - D(k,l)]^2 .$$

The total potential energy is

$$E = \sum E(i,j) .$$

In terms of the spring model, the ends of the level arms are now constrained to lie on a finite number of positions on the two surfaces. Although this system is finite, it is also high-dimensional, and the problem of finding minimal-energy states is still difficult because the number of possible states is vast (exponential in M^2). Furthermore, there is no reason to believe that E is convex, and therefore no reason to believe that a discrete iterative improvement algorithm would work.

3.1 Standard (Canonical) Annealing

Simulated annealing is a fairly new technique for solving such combinatorial optimization problems. In Section 3.3 a new variety of simulated annealing (called microcanonical annealing) is presented which has several advantages for computer implementation. In Section 3.1 the basic principles of the standard form of simulated annealing are described to set a context for the introduction of microcanonical annealing. An excellent review of simulated annealing may be found in Laarhoven and Aarts [9].

The most fundamental result of statistical physics is the Boltzmann (or Gibbs) distribution:

$$Pr(E_i) = \frac{\exp(-E_i/kT)}{Z(T)} ,$$

which gives the probability of finding a system in state i with energy E_i, assuming that the system is in equilibrium with a large heat bath at temperature kT (k is Boltzmann's constant). The normalizing quantity in the denominator, called the partition function, is a sum over all accessible states ν:

$$Z(T) = \sum_{\nu} \exp(-E_{\nu}/kT) . \tag{7}$$

Physicists are generally interested in calculating macroscopic properties of model systems at various temperatures. The average value of some macroscopic variable A (which may be the average energy of the system, for example) can be written:

$$\langle A \rangle = \sum_{\nu} A_{\nu} Pr(A_{\nu}) = \frac{\sum_{\nu} A_{\nu} \exp(-E_{\nu}/kT)}{Z(T)} .$$

1. Begin with the system in an arbitrary state ν.
2. Make a small change to the state, typically by changing the system in only one degree of freedom. Call the new state ν'.
3. Evaluate the resulting change in energy: $\Delta E = E_{\nu'} - E_{\nu}$.
4. If $\Delta E < 0$ (that is, the change takes the system to a state of lower energy) accept the change.
5. If $\Delta E \geq 0$ accept the change with probability $\exp(-\Delta E/kT)$.
6. Repeat steps (2) through (5) until the system reaches equilibrium.

Figure 2: The Metropolis algorithm.

Unfortunately, the partition function is usually impossible to calculate.

In 1953 Metropolis *et. al.* [8] described a Monte Carlo algorithm that generates a sequence of states which converges to the Boltzmann distribution in the limit (Figure 2). This method, which simulates the effect of allowing the system to interact with a much larger heat bath, samples what is called the canonical ensemble. Macroscopic parameters can then be calculated without knowledge of the partition function by averaging over long sequences, weighting each state by its Boltzmann factor $\exp(-E_{\nu}/kT)$. The Metropolis algorithm begins in a random state and then successively generates small, random state transitions ($\nu \rightarrow \nu'$) with the following probability:

$$Pr(\nu \rightarrow \nu') = \begin{cases} 1 & \text{if } \Delta E < 0 \\ \exp(-\Delta E/kT) & \text{otherwise} \end{cases} \tag{8}$$

where $\Delta E = E_{\nu'} - E_{\nu}$. Asymptotic convergence of the Metropolis algorithm to the Boltzmann distribution is guaranteed if the process for generating candidate state transitions is ergodic.

Kirkpatrick [10] and Černy [11] independently recognized a profound connection between the Metropolis technique and combinatorial optimization problems. If the energy of a state is considered as an objective function to be minimized, the minimum can be approximated by generating sequences at decreasing temperatures, until finally a ground state, or a state with energy very close to to a ground state, is reached at $kT = 0$. This is analogous to the physical process of annealing.

There are results showing the existence of annealing schedules (i.e., the rate of decrease of temperature) that guarantee convergence to ground states in finite time [12], but these schedules are too slow for practical use. Faster *ad hoc* schedules have been used in many problems with good average-case performance. While faster schedules may not find an optimal state, they can converge to states that are very close to optimal. The application of standard simulated annealing to the stereo matching problem is straightforward. An early version is described in [13]. (Marroquin [14] and Divko [15] have independently described similar methods.) In the following sections a more efficient version is presented.

3.2 Annealing over Scale

Simulated annealing could be applied directly to a pair of stereo images at the finest scale, but the convergence would be rather slow if the images had a large range of disparity. This was the approach reported in [13] (using the standard annealing algorithm and a slightly different energy function).

A more efficient method is to use the coarse-to-fine strategy that has been found to be so effective in other image-matching work. At a coarse level of resolution the number of lattice sites and the range of disparity are small; therefore, the size of the state space is relatively small. We should be able to compute an approximate ground state quickly, and then use it to initialize the annealing process at the next, finer level of resolution, and so on.

The laplacian pyramid, originally developed as a compact image-coding technique [16], offers an efficient representation for hierarchical annealing. In a laplacian pyramid an n x n image (for convenience, assume n is a power of 2) is transformed into a sequence of bandpass-filtered copies, $\{I_k, k = 0, \ldots, n\}$, where I_k is an image of size 2^{n-k} x 2^{n-k}. Each image is therefore smaller than its predecessor by a factor of 1/2 in linear dimension and a factor of 1/4 in area. We will refer to I_k as the image at level k. The center frequency of the passband is reduced by one octave between levels. This transform can be computed efficiently by recursively applying a small generating kernel to create a gaussian (low-passed) pyramid, and then differencing successive low-passed images to construct the laplacian pyramid. The difference-of-gaussians gives a good approximation to the $\nabla^2 G$ filter.

After constructing laplacian pyramids from the original stereo images, disparity is reduced by a factor of 1/2 in successive scales. Therefore, at some level, disparity is small everywhere. For typical stereo images, we can take this to be level $n-3$. (For example, if the original images were a power of 2 in linear dimension, the laplacian images at level $n-3$ would be 8 x 8 pixels. Disparities in the range of 0 to 63 pixels in a pair of 512 x 512 images would be reduced to 0, with truncation.) We shall start annealing at this level, find an approximate ground state, and then expand the solution to the next scale. To make this coarse-to-fine strategy work, however, we must specify how a low-resolution result is used to start the annealing process at the next-higher scale.

Expanding a low-resolution result to the next level presents a problem. Obviously, one should begin by simply doubling the size of the low-resolution lattice and doubling the disparity values. Having done this, however, the new state has a low energy but the system is not close to equilibrium. Every odd disparity value is "unoccupied," and the new map is therefore more uniform than it should be. This spurious uniformity, which is solely due to the quantization of the previous result, is likely to place the system near a local minimum from which it will not recover. Fortunately, there is an easy solution to this problem: destroy this uniformity by adding heat. Simply run the annealing algorithm "in reverse" by adding energy instead of removing it. Heating may proceed much faster than cooling because the system relaxes to equilibrium quickly at high temperatures.

3.3 Microcanonical Annealing

1. Begin with the system in an arbitrary state ν.
2. Make a small change to the state, typically by changing the system in only one degree of freedom. Call the new state ν'.
3. Evaluate the resulting change in energy: $\Delta E = E_{\nu'} - E_{\nu}$.
4. If $\Delta E < 0$ accept the change and increase the demon energy $(E_D \leftarrow E_D - \Delta E)$.
5. If $\Delta E \geq 0$ accept the change contingent upon E_D:
    - If $\Delta E < E_D$ accept the change and decrease the demon energy $(E_D \leftarrow E_D - \Delta E)$.
    - Otherwise, reject the change.
6. Repeat steps (2) through (5) until the system reaches equilibrium.

Figure 3: The Creutz algorithm.

Creutz has described an interesting alternative to the Metropolis algorithm[17], outlined in Figure 3. Instead of simulating the effect of a large heat bath, the Creutz algorithm simulates a thermally isolated system in which energy is conserved. Samples are drawn from the microcanonical ensemble. One can imagine the difference between the Metropolis algorithm and the Creutz algorithm as follows. The Metropolis algorithm generates a "cloud" of states, each with, in general, different energies, which fills a volume of phase space. As temperature decreases this volume contracts to one or more ground states. The Creutz algorithm, by contrast, generates states on a constant-energy surface in a somewhat larger phase space. As energy decreases these surfaces shrink to the same set of ground states.

The simplest way to accomplish this is to augment the system with one additional degree of freedom, called a demon, which carries a variable amount of energy, E_D. This demon holds the kinetic energy of the system and, in effect, replaces the heat bath. The total energy of the system is now

$$\begin{aligned} E_{total} &= E_{potential} + E_{kinetic} \\ &= E + E_D \end{aligned}$$

The demon energy, being kinetic, is constrained to be nonnegative. The algorithm accepts all transitions to lower energy states, adding $-\Delta E$ (the energy given up) to E_D. Transitions to higher energy are accepted only when $\Delta E < E_D$, and the energy gained is taken away from E_D. Total energy remains constant.

Microcanonical annealing simply replaces the Metropolis algorithm with the Creutz algorithm. Instead of explicitly reducing temperature, the microcanonical annealing algorithm reduces energy by gradually lowering the value of E_D. Standard arguments can be used to show that at equilibrium E_D assumes a Boltmann distribution over time [17]:

$$Pr(E_D = E) \propto \exp(-E/kT) \; .$$

Temperature therefore emerges as a statistical feature of the system:

$$kT = \langle E_D \rangle \; . \tag{9}$$

This simple version of microcanonical annealing, using only one demon, is not suited to parallel implementation. Each decision to accept or reject a state transition depends on the value of E_D, and therefore on the previous decision. The computation can be made parallel by using a lattice of demons. Temperature is still measured with (9), but using the distribution of E_D over space rather than time.[2]

There is a minor complication in using a lattice of demons. The single-demon algorithm visits sites at random and the demon allows energy to be transferred throughout the lattice. Similarly, in the lattice-of-demons algorithm the demons must be mixed throughout the lattice. If this is not done energy will be transferred between sites very slowly, *only through the nearest-neighbor interactions* of (6). We use a complete random permutation of the demons after every lattice update, but more local methods are also adequate.

Microcanonical annealing has several advantages over standard annealing:

1. It does not require the evaluation of the transcendental function $\exp(x)$. Of course, in practice this function can be stored in a table, but we would like our algorithm to be suited to fine-grained cellular automaton with very limited local memory.

2. It is easily implemented with low-precision integer arithmetic; again, a significant advantage for simple hardware implementation.

3. In the Metropolis algorithm a state transition is accepted or rejected by comparing $\exp(-\Delta E/kT)$ to a random number drawn from a uniform distribution over $[0,1]$, and these numbers must be accurate to high precision. The Creutz algorithm does not require high-quality random numbers.

Experiments indicate that the Creutz method can be programmed to run an order of magnitude faster than the conventional Metropolis method for discrete systems [18].

In standard annealing it is not clear how to determine when the system reaches equilibrium. One can examine fluctuations in the average energy, which should be of order $1/M^2$ at equilibrium, but this may require many extra iterations to get adequate statistics because one does not know in advance what average value to expect. In microcanonical annealing there is a simpler way. Let r_{eq} be the ratio of the observed average demon energy to the standard deviation of the same observed distribution:

$$r_{eq} = \frac{\langle E_D \rangle}{\sigma(E_D)} \; . \qquad (10)$$

At equilibrium $r_{eq} \approx 1$.

As with the Metropolis algorithm, the Creutz algorithm converges to the Boltzmann distribution in the limit for any ergodic process generating candidate state transitions. Of course, different state-transitions schemes will affect the rate of convergence. We have found the following simple method to be adequate:

$$Pr(d \rightarrow d') = \begin{cases} .5 & \text{if } |d - d'| = 1 \\ 0 & \text{otherwise} \end{cases}$$

In other words, the disparities increase or decrease by one lattice position as the system follows a Brownian path on its phase-space surface of constant energy. Only one random bit need be generated for each transition.

4 Experimental Results

This section presents experimental results for three distinct cases: a sparse random-dot stereogram, a high-resolution aerial stereo pair, and a medium-resolution, oblique, ground-level scene with *prominent* occlusions. The method has been tested on over 30 real images; these examples have been chosen to indicate a variety of conditions. Identical parameters were used for all three cases. Four nearest neighbors were used for $\mathcal{N}$. We used a value of 50 for λ, which works well for images quantized into eight-bit values. A schedule for heating and cooling was established to yield about 400 complete scans at each scale, with about 90 percent of the cycle devoted to cooling. The precise number of scans varies because during cooling the system adapts the schedule to stay near equilibrium.

Figure 4 shows the results for a 10% random-dot stereogram with four depth planes separated by intervals of 2 pixels of disparity. The three graphs trace the evolution of temperature, average energy per lattice site, and r_{eq}. Note that the plot of r_{eq} indicates that the system moves away from equilibrium during the relatively fast heating cycles, but relaxes quickly back to equilibrium after cooling starts. The system appears to drop away from equilibrium at low temperatures according to the r_{eq} plot, but this effect is actually because there are very few energy levels available to the demons near the ground state.

Figure 5 shows results from a 512 x 512 aerial stereo pair with a disparity range of 72 pixels. This is the largest problem we have attempted so far. The disparity map, shown in the lower left, is also shown to the right with contours at every fifth disparity level. Inspection of this map indicates that it is accurate to 1 pixel over the entire field.

The stereo pair in Figure 6 is distinguished by sharp discontinuities of depth. (Contours are shown at every third disparity level.) The method has done a reasonably good job of separating the tree from the background, which is somewhat surprising since it has no explicit representation for occlusions. In an attempt to model occlusions we have experimented with line processes, like the one used by Geman and Geman [12], and with non-linear "springs" that weaken as they deform. Our results so far have not justified the added complexity.

In earlier work [13,19] we used the absolute difference instead of the squared difference in (3). This is slightly more efficient to implement and makes little difference in the results. In terms of the spring model, this would correspond to using "springs" that exert a constant force in opposite direction of their deformation. Of course, a different value of λ is required. We have also experimented with eight-neighbor versions of $\mathcal{N}$, but no significant improvement in performance was observed.

5 Conclusions

The major conclusion we can draw is that (1) is an adequate criterion for stereo matching, even in scenes with abrupt occlusions. The results in Figure 6 indicate that solutions can accommodate very abrupt changes in depth. Residual energy in the near-optimal states is concentrated along steep disparity contours rather than spread over large areas. The cyclopean representation ensures that there will be no "unseen" areas with undefined disparity.

The use of a scale hierarchy dramatically increases the efficiency of the method, especially for large problems such as that

[2] Statistics can be sampled over both time and space, if desired.

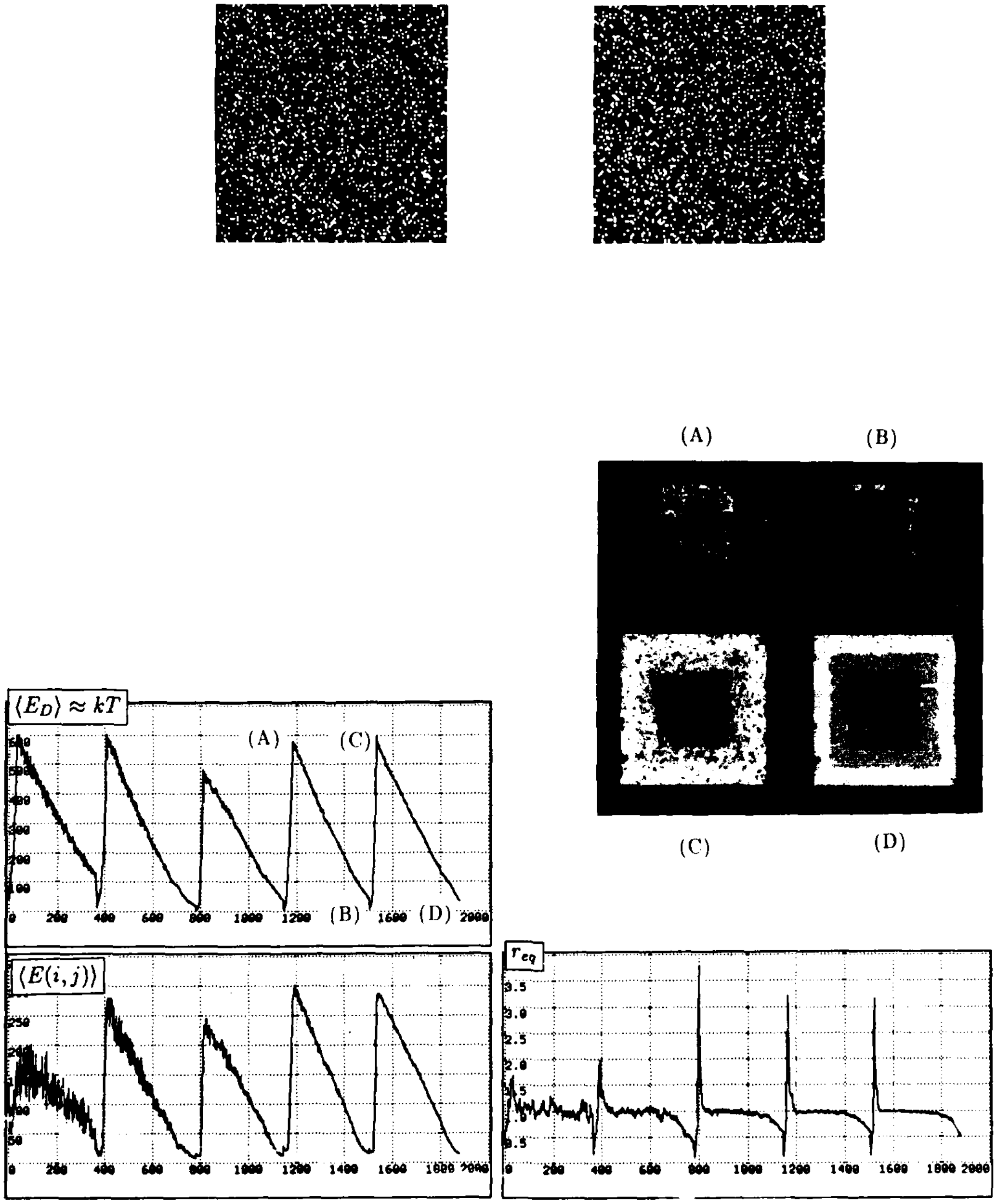


Figure 4: A 10% random-dot stereogram.

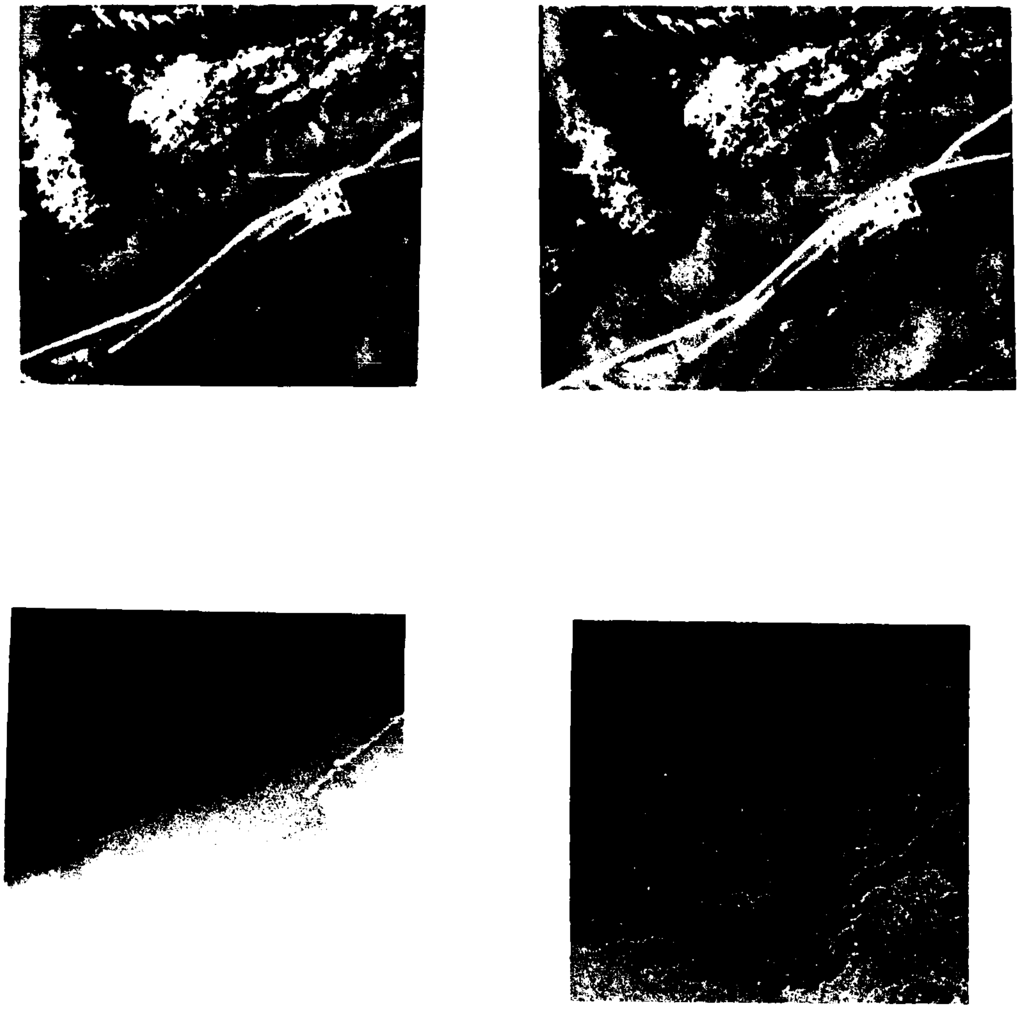


Figure 5: A high-resolution aerial stereogram.

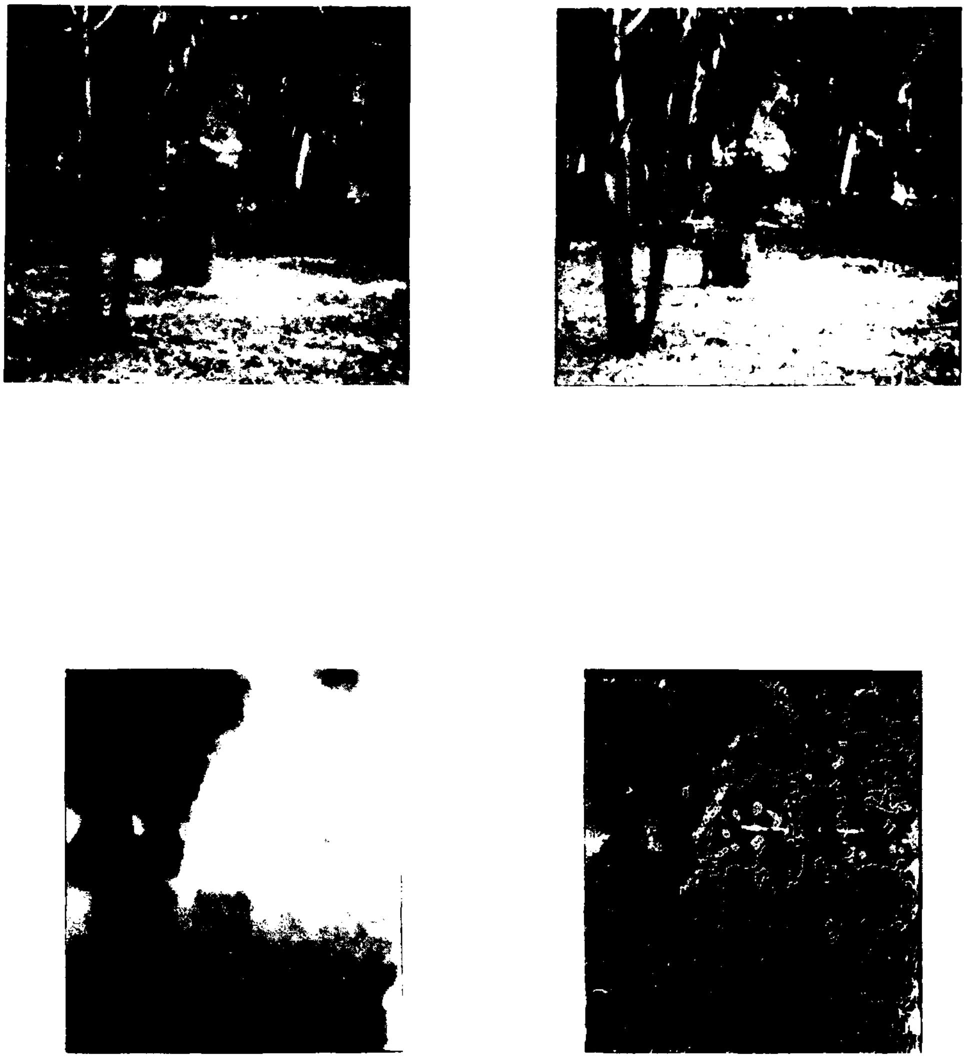


Figure 6: An oblique, ground-level stereogram with occlusion.

illustrated in Figure 5. An additional benefit of using a scale hierarchy is that the solution is less sensitive to small amounts of vertical disparity, which is eliminated at coarser scales . (Uncertainty in the camera model will usually cause some vertical disparity in high-resolution images.) A gaussian low-pass hierarchy works as well as the laplacian hierarchy if the images are recorded with equivalent sensors. The benefit of bandpass filtering is to eliminate the low-frequency variation caused by uncalibrated photometry.

Annealing provides a way to bridge the gap between scales. The microcanonical annealing algorithm appears to be an improvement over canonical annealing for reasons discussed in Section 3.3. It is certainly much easier to implement in cellular automata. The theoretical results showing convergence in finite time do not necessarily carry over to microcanonical annealing, but the requirements of these results are never met in practice anyway.

Canonical annealing and "pure" single-demon microcanonical annealing are at opposite ends of a spectrum. In canonical annealing the heat bath is much larger than the model system, and is not represented explicitly. In pure microcanonical annealing the heat bath — that is, the single demon — is much smaller that the system, and it is represented explicitly. The lattice-of-demons algorithm is midway between these extremes, with the heat bath and the model system having comparable sizes. In a sense, this is a classical space/time tradeoff. By representing the heat bath explicitly we can avoid the evaluation of complicated functions.

Comparison with the scale-space continuation method is difficult because the nature of the data affects the smoothness of the energy landscape. In some stereo pairs the data will be so clear that this more direct form of optimization will work well. (For example, dense, random, greyscale stereograms with intensities chosen over a broad range of values can be solved even by a "greedy" algorithm; that is, a Monte Carlo optimization accepting only transitions that lower energy.) Both methods use multi-scale representations, but the stochastic approach uses multiple scales for efficiency, while the gradient descent method uses them in an attempt to impose convexity. A comparative study is needed to determine when the additional overhead of annealing is justified.

References

[1] G. Sperling, "Binocular vision: A physical and neural theory," J. AM. PSYCHOL., vol. 83, pp. 461-534, 1970.

[2] B. Julesz, FOUNDATIONS OF CYCLOPEAN PERCEPTION, Univ. of Chicago Press: Chicago, Ill., 1971.

[3] D. Marr and T. Poggio, "Cooperative computation of stereo disparity," SCIENCE, vol. 194, pp. 283-287, 1976.

[4] T. Poggio, V. Torre, and C. Koch, "Computational vision and regularization theory," NATURE, vol. 317, pp. 314-319, 1985.

[5] A. Witkin, D. Terzopoulos, and M. Kass, "Signal matching through scale space," INTERNATIONAL JOURNAL OF COMPUTER VISION, vol. 1, pp. 133-144, 1987.

[6] B.K.P. Horn, ROBOT VISION, M.I.T. Press: Cambridge, MA, 1986.

[7] G.H. Walker and J. Ford, "Amplitude instability and ergodic behavior for conservative nonlinear oscillator systems," PHYS. REV. vol. 188, pp. 416-432, 1969.

[8] N. Metropolis, A.W. Rosenbluth, M.N. Rosenbluth, A.H. Teller, and E. Teller, "Equations of state calculations by fast computing machines," J. CHEM. PHYS., vol. 21, pp. 1087-1092, 1953.

[9] P.J.M. van Laarhoven and E H.L. Aarts, SIMULATED ANNEALING: THEORY AND APPLICATIONS, D. Reidel Publishing Co.: Dordrecht, Holland, 1987.

[10] S. Kirkpatrick, C.D. Gelatt, and M.P. Vecchi, "Optimization by simulated annealing," SCIENCE, vol. 220, pp. 671-680, 1983.

[11] V. Černy, "Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm," J. OPT. THEORY APPL., vol. 45 41-51, 1985.

[12] S. Geman and D. Geman, "Stochastic relaxation, Gibbs distributions, and Bayesian restoration of images," IEEE TRANS. PAMI, vol. PAMI-6, pp. 721-741, 1984.

[13] S. Barnard, "A stochastic approach to stereo vision," in PROC. 10TH INT. JOINT CONF. ARTIF INTELL., Milan, Italy, 1987, pp. 832-835.

[14] J.L. Marroquin, "Probabilistic solution of inverse problems," M.I.T. Artif. Intell. Lab., Cambridge, MA, A.I.-TR 860, 1985.

[15] R. Divko and K. Schulten, "Stochastic spin models for pattern recognition," Physik-Department, Technische Universität München, personal correspondence, 1987.

[16] P. Burt, "The laplacian pyramid as a compact image code," IEEE TRANS. COMMUNICATIONS, vol. COM-31, pp. 532-540, 1983.

[17] M. Creutz, "Microcanonical Monte Carlo simulation," PHYSICAL REV. LET., vol. 50, pp. 1411-1414, 1983.

[18] G. Bhanot, M. Creutz, and H. Neuberger, "Microcanonical simulation of Ising systems," NUCLEAR PHYSICS, B235[FS11], pp. 417-434, 1984.

[19] S. Barnard, "Stereo matching by hierarchical, microcanonical annealing," in PROC. NATL. CONF. ARTIF. INTELL AAAI-86, Philadelphia, PA, 1986, pp. 676-680.

QUALITATIVE VS. QUANTITATIVE DEPTH AND SHAPE FROM STEREO

Daphna Weinshall

CBIP, E25-201, MIT, Cambridge MA 02144

ABSTRACT

This paper concentrates on the problem of obtaining depth information from binocular disparities. It is motivated by the fact that implementing registration algorithms and using the results for depth computations is hard in practice with real images due to noise and quantization errors. We will show that qualitative depth information can be obtained from *stereo disparities with almost no computations*, and with no prior knowledge (or computation) of camera parameters. The only constraint is that the epipolar plane of the fixation point includes the X−axes of both cameras. We derive two expressions which order all matched points in the images in two distinct depth-consistent fashions from image coordinates only. One is a tilt-related order λ, which depends only on the polar angles of the matched points, the other is a depth-related order χ. Using λ for tilt estimation and point separation (in depth) demonstrates some anomalies and unusual characteristics which have been observed in psychophysical experiments. Furthermore, the same approach can be applied to estimate some qualitative behavior of the normal to the surface of any object in the field of view. More specifically, one can follow changes in the curvature of a contour on the surface of an object, with either x- or y-coordinate fixed.

INTRODUCTION

Research in early vision regarding stereo seems to be concerned mainly with the correspondence problem, namely, finding the right matching between points on the left and right images. Obtaining exact depth values from a stereo pair has been considered a simple exercise, whose solution is well known, though might involve some tedious but trivial computations. Thus, it has been implicitly assumed that the final goal of stereo algorithms is to compute exact depth map using disparity values. The following observations suggest, however, that depth computation from disparity values is *not necessarily straightforward* or even feasible, and that more qualitative depth information may be easier to obtain and more robust.

First, the depth computation problem reduces to a simple trigonometric formula when the parameters of the cameras, or the eyes, are known. When they are not known, a scheme to compute camera's parameters from a number of conjugate points (that is, matched pairs of points from the different images) has been devised, involving the solution of a set of nonlinear equations (see for instance Horn, 1986). Since the problem has no closed-form solution, and since the data are not precise, a solution is found using iterative methods that minimize the sum of the squares of the errors. In practice, however, this approach is very difficult to implement, since the parameters of the cameras must be obtained from data with error in the order of magnitude of the disparity values, which are the raw material used for depth computation (e.g., error due to pixel quantization). In other words, the registration problem (namely, finding parameters of cameras calibration) is much more difficult than just computing depth from disparity values. Less general methods to find camera calibration have also been devised, see Prazdny (1981) and Longuet-Higgins (1981).

The other observation originates from biological vision. It seems that human vision does not necessarily obtain exact depth values from stereo disparity information alone, see, e.g., Foley (1977) and Foley & Richards (1972). Rather, stereo disparity seems to be used mainly in obtaining qualitative depth information about objects in the field of view. Estimation of the magnitude of this relative depth is possibly dependent on extraretinal estimation of some physical parameters like angle of convergence of the eyes. For example, whether looking at stereograms with crossed or uncrossed eyes affects only the extraretinal perception of the angle of convergence of the eyes, not the disparity values. It also results in a different perception of the depth of the central square in a simple random-dot stereogram (where a central square in one image has a constant shift with respect to the other).

The purpose of this paper is to exploit the geometry of the situation, where a scene is viewed from two different angles, to get insight into the above problem. It will be shown that qualitative relative depth information (order) of various kinds can be obtained from conjugate points alone easily and reliably, involving almost no computations and independent of camera's parameters. These orders will demonstrate some anomalies which are observed in human psychophysics and presently lack other straightforward explanation.

The exact order expressions will scale in proportion to the angle of convergence between the two cameras. The exact relative depth can be computed from these orders using few matched points and some approximated numeric scheme, or using more than two images. Alternatively, it can be estimated from some external estimation of the physical quantities involved, namely - the angle of convergence and the angle of gaze, in agreement with the psychophysical theory suggested in Foley (1977).

BASIC GEOMETRY

Given two cameras, assume that the optical axes intersect at the fixation point. Also, assume that the epipolar plane of the fixation point (the plane through the optical axes of both cameras and their baseline, henceforth "base plane") includes the X-axes of both cameras (which are, therefore, epipolar lines by *definition*). Let us define the following coordinate system (see figure 1): let the fixation point be the origin, the base plane (which passes through this point) be the $X - Z$ plane, and the line perpendicular to this plane through the origin be the Y axis. On the $X - Z$ plane, the optical axes of both cameras intersect in the origin and create an angle 2μ between them. Let the Z axis be the angle bisector of 2μ, and the X axis perpendicular to the Z axis in the $X - Z$ plane. This system is very similar to the cyclopean coordinate system commonly used in the literature, with the exceptions that the angle bisector is replaced by the median to the interocular line and the origin is translated to the mid point of the interocular line. A similar system can be defined for the case of motion, if the fixation point is kept constant. That is, the observer follow the same point with his eyes. This is more typical of human vision than machine vision.

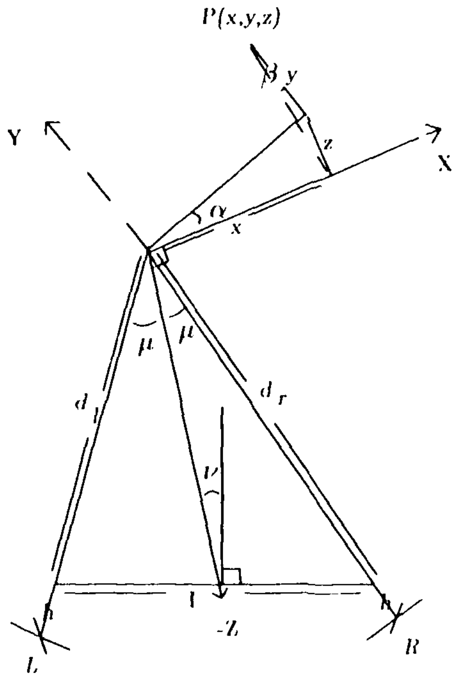


Figure 1 The base plane ($X - Z$) viewed from above, with two cameras

For a given point $P = (x, y, z)$, *let α denote the angle of tilt* — the polar angle of its projection on the $X - Z$ plane ($\alpha = \arctan(\frac{z}{x})$). *Let β denote the angle of slant* — the polar angle of its projection on the Y Z plane ($\beta = \arctan(\frac{z}{y})$). Thus P can be also written as $P = (\frac{z}{\tan\alpha}, \frac{z}{\tan\beta}, z)$, where z is its depth relative to the fixation point in the above coordinate system. Let (x_l, y_l) and (x_r, y_r) be the Cartesian coordinates of the projection of P on the left *and right images respectively*. Using polar coordinates, the two projections can be written as (r_l, θ_l) and (r_r, θ_r) respectively. Let $\lambda = \frac{\cot\theta_r}{\cot\theta_l}$. Then the following can be shown to hold (see appendix):

$$\tan\alpha = \frac{1}{\tan\mu} \cdot \frac{\lambda - 1}{\lambda + 1} \tag{1}$$

$$\tan\beta = \frac{\cot\theta_r - \cot\theta_l}{2\sin\mu} \tag{2}$$

Thus, the two angles α and β depend only on the angle of convergence and the polar angles of the conjugate points. *It can be shown* that the polar angles are preserved under projection, through any point on the optical axis, onto either a spherical body (like the eye) or a planar one (a camera). There is no dependence on other parameters of the cameras (which could be different ones), their relative positions, or the angle of gaze. Equation (1) will be used in the next section to obtain an order on all matched image points in each visual hemifield according to *their tilt*. *This order is parameters* independent and demonstrates psychophysical anomalies which presently lack other straightforward explanation. Equation (2) will be used to obtain an expression for the relative depth z. However, this expression will depend on cameras' parameters like focal length and interocular distance, and the angle of gaze. A parameter-independent relative depth order can be obtained from this expression. *It will hold for small angle of convergence* 2μ, *as will* be discussed in a succeeding section.

TILT-RELATED ORDER

From (1) it immediately follows that α is a monotonic increasing function of λ for a fixed configuration of the cameras. Thus, the ratio $\frac{\lambda_i}{\lambda_j} = \frac{\cot\theta_r^i}{\cot\theta_l^i} / \frac{\cot\theta_r^j}{\cot\theta_l^j}$ gives qualitative relative distance information on any two points i and j in each visual hemifield in the following sense: if the ratio is greater than 1, meaning $\alpha_i > \alpha_j$, then a separating plane between points i and j through the fixation point and perpendicular to the base plane will leave point j and the viewer on one of its sides, and point i on the other side "further away" from the viewer. *In other words*, λ defines an order on all the matched points in a given hemifield. This order corresponds to Euclidean distance if points i and j are approximately on the same line of sight from the viewer, namely — about the same image x coordinate.

Note that if λ is constant on all points segmented as belonging to the central object, then it is a planar object with some tilt towards or away from the viewer, according to the sign of $\lambda - 1$. Moreover, since

$$\theta_l \gtrless \theta_r \Leftrightarrow \frac{\cot\theta_r}{\cot\theta_l} \gtrless 1 \Leftrightarrow \alpha \gtrless 0°,$$

it follows that $\theta_l - \theta_r$ also gives a qualitative estimation to the tilt of a point P relative to the fixation point. If the fixation point is at the same distance from both cameras, this estimate would indicate whether P, relative to the fixation point, is tilted away from the cameras' baseline ($\alpha > 0^o$), "parallel" to the baseline ($\alpha \approx 0^o$), or towards the cameras' baseline ($\alpha < 0^o$).

The order expression λ has been defined as a function of the polar angle ϑ only in both eyes. This is especially convenient since the polar angle is preserved under projection onto either a spherical body (the eye) or a planar body (a camera). However, it might prove useful to examine λ as a function of the Cartesian coordinates (x_l, y_l) and (x_r, y_r) in both images, assuming planar projection. In this case:

$$\begin{aligned} \ln \lambda = \ln \frac{\cot \vartheta_r}{\cot \vartheta_l} &= \ln \frac{x_r/x_l}{y_r/y_l} \\ &= (\ln x_r - \ln x_l) - (\ln y_r - \ln y_l) \\ &= \Delta(\ln x) - \Delta(\ln y). \end{aligned} \qquad (4)$$

In other words, if any matching algorithm is applied to the output images of the transformation $T : (x, y) \longrightarrow (\ln x, \ln y)$ performed on the original images, and the disparity vector (Δ_x, Δ_y) is then computed in the usual way, then the difference $\Delta_x - \Delta_y = \ln \lambda$ is an order of the same type as λ, with no need for any additional computation. However, the transformation T is singular near the vertical and horizontal meridians.

An interesting feature of the order λ is its good agreement with some characteristics of human vision, especially under the unrealistic conditions of the "induced-effect" which is predicted by using λ (see Weinshall, 1987).

DEPTH-RELATED ORDER

From equation (2) one can obtain an explicit expression for the depth z of a point relative to the fixation point (the origin). First, note that (2) implies

$$z = (\cot \vartheta_r - \cot \vartheta_l) \cdot \frac{y}{2 \sin \mu}. \qquad (5)$$

Thus, $\lambda^y = (\cot \vartheta_r - \cot \vartheta_l)$ gives exact depth order on all the points in space with some constant height y over the base plane. *It follows that this order is most useful to compare points which* differ mainly in their x-coordinate with y approximately the same. The previous order λ, on the other hand, was most useful to compare points which differed mainly in their y coordinate with x approximately the same.

Next, let us derive an expression for z which depends only on scene and camera's parameters. In the appendix it is shown that, for ν the angle of gaze, I- the interocular distance and, h- the focal length of the cameras,

$$y = \left\{ \frac{I \cos(\mu - \nu)}{\sin 2\mu} - x \sin \mu + z \cos \mu \right\} \cdot \frac{y_r}{h}, \qquad (6)$$

or

$$y = \left\{ \frac{I \cos(\mu + \nu)}{\sin 2\mu} + x \sin \mu + z \cos \mu \right\} \cdot \frac{y_l}{h} . \qquad (6')$$

Substituting (6) in (5) for a point in the right hemifield, ((6′) will be used otherwise), gives:

$$z = \frac{I \cos(\mu - \nu) / \sin 2\mu}{\left\{ \frac{\sin \mu}{x_r - \frac{y_r}{y_l} x_l} \left[2h + \tan \mu (x_r + \frac{y_r}{y_l} x_l) \right] - \cos \mu \right\}}.$$

Thus, for an angle of *convergence* 2μ *small enough so that* $2h \gg |\tan \mu (x_r + \frac{y_r}{y_l} x_l)|$, we get a relative depth order χ on all the points in the visual field, where

$$\chi = x_r - \frac{y_r}{y_l} x_l.$$

As will be shown in the section of error analysis, $\frac{y_r}{y_l} = 1 + O(\mu)$. Likewise, since the field of view is mechanically bounded by some $2\xi < 180^o$, it follows that $x < h \tan \xi$. Thus, a sufficient condition for the appropriateness of χ, *to a first order in* μ, is $1 \gg \tan \mu \cdot \tan \xi$. If $2\xi \leq 90^o$, which is a reasonable upper bound, then it is sufficient if $1 \gg \tan \mu$, or $2\mu \ll 90^o$. To illustrate, the distance to the point of fixation should be much greater than 3 cm for an average person looking straight ahead, possibly 30 cm or more.

We have got, then, a relative depth order which is the traditional x-disparity corrected for non-zero vergence (angle of gaze ν not 0) and some field location (x-coordinate) distortion. However, for a fixed convergence angle 2μ, this order has some distortion relative to the physical relative depth, which increases with the horizontal distance from the point of fixation (the x-coordinate).

QUALITATIVE SHAPE FROM STEREO

The triple (α, β, z) as *point representation*, and *equations* (1) and (2), turn out to be useful for surface normal analysis. For any two points P_1 and P_2, where $\vec{P}_1 = z_1(\frac{1}{\tan \alpha_1}, \frac{1}{\tan \beta_1}, 1)$ and $\vec{P}_2 = z_2(\frac{1}{\tan \alpha_2}, \frac{1}{\tan \beta_2}, 1)$, *let* $\vec{N} = \vec{P}_1 \times \vec{P}_2$. $\vec{N}$ *is perpendicular* to $(\vec{P}_1 - \vec{P}_2)$. (It is actually proportional to the normal to the plane passing through P_1, P_2, and the fixation point.) After *some calculations, it can be shown that*

$$\vec{N} = z(\frac{\cot \beta_1 - \cot \beta_2}{\cot \alpha_1 \cot \beta_2 - \cot \alpha_2 \cot \beta_1}, \frac{\cot \alpha_2 - \cot \alpha_1}{\cot \alpha_1 \cot \beta_2 - \cot \alpha_2 \cot \beta_1}, 1)$$

$$= z(\frac{1}{\tan \mu} f(\vartheta_l^1, \vartheta_r^1, \vartheta_l^2, \vartheta_r^2), \frac{1}{\sin \mu} g(\vartheta_l^1, \vartheta_r^1, \vartheta_l^2, \vartheta_r^2), 1),$$

where

$$f(\vartheta_l^1, \vartheta_r^1, \vartheta_l^2, \vartheta_r^2) = \frac{(\cot \vartheta_l^2 - \cot \vartheta_l^1) - (\cot \vartheta_r^2 - \cot \vartheta_r^1)}{(\cot \vartheta_r^2 - \cot \vartheta_r^1) + (\cot \vartheta_l^2 - \cot \vartheta_l^1)}$$

$$g(\vartheta_l^1, \vartheta_r^1, \vartheta_l^2, \vartheta_r^2) = \frac{\cot \vartheta_r^2 \cot \vartheta_l^1 - \cot \vartheta_l^2 \cot \vartheta_r^1}{(\cot \vartheta_r^2 - \cot \vartheta_r^1) + (\cot \vartheta_l^2 - \cot \vartheta_l^1)}.$$

Thus, as long as $f(\vartheta_l^1, \vartheta_r^1, \vartheta_l^2, \vartheta_r^2)$ and $g(\vartheta_l^1, \vartheta_r^1, \vartheta_l^2, \vartheta_r^2)$ remain constant, which can be determined from image coordinates only, the points are coplanar (among themselves and with the fixation point), or the object at the center of gaze is planar. Note that λ is obtained from f when $\cot \vartheta_l^2 = \cot \vartheta_r^2 = 0$ ($g = 0$ then).

Moreover, for any object it is possible to obtain qualitative information about its surface along any contour, with either x or y fixed. Take a contour on the surface with some fixed y coordinate, and let P_1 and P_2 be two points on it. Since the y-coordinate of $\vec{P}_1 - \vec{P}_2$ is 0, the projection of $\vec{N}$ on the X - Z plane, $\vec{n} = z(\frac{1}{\tan \mu} f(\vartheta_l^1, \vartheta_r^1, \vartheta_l^2, \vartheta_r^2), 1)$, is perpendicular to the projection of $\vec{P}_1 - \vec{P}_2$. Thus, for fixed y, the one dimensional boundary contour is convex when $f(\vartheta_l^1, \vartheta_r^1, \vartheta_l^2, \vartheta_r^2)$ increases with increasing x, concave when $f(\vartheta_l^1, \vartheta_r^1, \vartheta_l^2, \vartheta_r^2)$ decreases, and linear when $f(\vartheta_l^1, \vartheta_r^1, \vartheta_l^2, \vartheta_r^2)$ remains constant. Note that χ^y can be obtained from f since the sign of f determines relative depth between two points with fixed y-coordinate. The same qualitative description can be obtained for any boundary contour with fixed x from following $g(\vartheta_l^1, \vartheta_r^1, \vartheta_l^2, \vartheta_r^2)$ with increasing y, x fixed. This qualitative description depends on image coordinates only, more specifically on polar angles of the conjugate points. Obtaining this description is not trivial, though, since such a contour in the world coordinate system will be usually mapped to an oblique line in the image plane due to convergence.

In the general case, the normal to a plane passing through three points in space P_1, P_2, and P_3 depends on the image coordinates and the angle of convergence μ in a more complex way, so that μ should be known to compute the (exact) 3-D normal. (The focal length h should be known as well.) However, if $\vec{P}_1 - \vec{P}_2 = (0, y, z)$ and $\vec{P}_2 - \vec{P}_3 = (x, 0, z)$ or vice versa, the normal can be computed from the above argument (up to a scaling factor of the x- and y-coordinates, depending on μ).

Alternatively, one can estimate the normal to the plane passing between three points P_1, P_2, and P_3 to a first order in μ and the x-disparity $\frac{x_r - \frac{y_r}{y_l} x_l}{2h \sin \mu}$. In this case, after substituting $K_{\mu\nu h} \cdot (x_r - \frac{y_r}{y_l} x_l)$ as an approximation for z, where $K_{\mu\nu h}$ is some constant which depends on μ, ν, and h, one gets an expression for the general normal $\vec{N}_G$:

$$\begin{aligned}\vec{N}_G &= (\vec{P}_1 - \vec{P}_2) \times (\vec{P}_2 - \vec{P}_3)\\ &\approx W_{\mu\nu h}(V_{\mu\nu h} F(x_r^1, x_l^1, y_r^1, y_l^1, x_r^2, x_l^2, y_r^2, y_l^2, x_r^3, x_l^3, y_r^3, y_l^3),\\ &\quad U_{\mu\nu h} G(x_r^1, x_l^1, y_r^1, y_l^1, x_r^2, x_l^2, y_r^2, y_l^2, x_r^3, x_l^3, y_r^3, y_l^3),\ 1) \quad .\end{aligned}$$

where $W_{\mu\nu h}$, $V_{\mu\nu h}$, and $U_{\mu\nu h}$ are some constants which depend on μ, ν and h. $F()$ and $G()$ are some functions of images coordinates only. Once again, one can verify planarity of surfaces of objects in the field of view when F and G remain constant.

NUMERICAL COMPUTATION

Let us compute the exact tilt and depth value to a first order in the convergence angle 2μ, following Mayhew & Longuet-Higgins (1982) method to compute tilt and slant of a plane through the fixation point. The following scheme, however, will be simpler and involve less and more rigorous assumptions (we shall only assume small 2μ as implied above). Since, to a first order in μ, $\tan 2\mu \approx \frac{I \cos(\nu)}{R}$, where R is the distance between the fixation-point and the midpoint of the interocular line (the nose), our computations will be to a first order in $(\frac{1}{R})$.

Let (x, y) and (x', y') denote the image coordinates of a certain point in space on the two cameras respectively. Let $\hat{\alpha}$ and $\hat{\beta}$ denote the parameters of a plane that passes through a given point in space and the fixation-point in the above coordinate system, so that $Z = \hat{\alpha}X + \hat{\beta}Y$. Thus $\hat{\alpha}$ is $\tan(\alpha)$ in the previous notations if $\hat{\beta} = 0$ and $\hat{\beta}$ is $\tan(\beta)$ if $\hat{\alpha} = 0$.

Then, to a first order in μ, we have (Longuet-Higgins & Prazdny, 1980)

$$\begin{aligned}\Delta x = x' - x &= [(\hat{\alpha}\cos(\nu) + \sin(\nu))x + \hat{\beta}\cos(\nu)y + (\cos(\nu)\\ &\quad - \hat{\alpha}\sin(\nu))x^2 - \hat{\beta}\sin(\nu)xy] \cdot I/R,\\ \Delta y = y' - y &= [\sin(\nu)y + (\cos(\nu) - \hat{\alpha}\sin(\nu))xy\\ &\quad - \hat{\beta}\sin(\nu)y^2] \cdot I/R.\end{aligned} \tag{7}$$

(The coordinate system used to obtain (7) is the cyclopean coordinate system. This, however, doesn't change the results when changing to our coordinate system since the angle bisector and the median are the same line to a first order in μ and the translation of the origin has been taken into account in the definition of the target plane).

For a given point in space, one can, in the more interesting cases, take the plane passing through it and the fixation point which is perpendicular to the base plane, for which $\hat{\beta} = 0$. This plane would be determined only by $\hat{\alpha}$ (the plane perpendicular to the base plane, unique usually.) Thus, we have

$$\begin{aligned}\frac{\Delta y}{y} &= [\sin(\nu) + (\cos(\nu) - \hat{\alpha}\sin(\nu))x] \cdot I/R\\ &= [\tan(\nu) + (1 - \hat{\alpha}\tan(\nu))x] \cdot \tan(2\mu).\end{aligned} \tag{8}$$

Let (x_1 , y_1 , Δx_1 , Δy_1) be the coordinates of a point on the vertical meridian, so that $x_1 \approx 0$. Then we have

$$\frac{\Delta y_1}{y_1} = \tan(\nu) \cdot \tan(2\mu).$$

(Recall that $\frac{1 - \frac{y_1'}{y_1}}{1 + \frac{y_1'}{y_1}} = \tan(\nu) \cdot \tan(\mu)$ always.)

Let (x_2 , y_2 , Δx_2 , Δy_2) be the coordinates of a *point* with $\hat{\alpha} \approx 0$. Such a point, if exists, can be easily identified since it satisfies $\frac{y'}{y} \approx \frac{x'}{x}$. Then we have:

$$x_2 \cdot \tan(2\mu) = \frac{\Delta y_2}{y_2} - \tan(\nu) \cdot \tan(2\mu) = \frac{\Delta y_2}{y_2} - \frac{\Delta y_1}{y_1} = \frac{y_2'}{y_2} - \frac{y_1'}{y_1}.$$

In other words,

$$\tan(2\mu) = \frac{1}{x_2} \cdot [\frac{y_2'}{y_2} - \frac{y_1'}{y_1}].$$

Now, for any point (x, y) in the image we have, using (7) with $\hat{\beta} = 0$:

$$\frac{x'}{x} - \frac{y'}{y} = \frac{\Delta x}{x} - \frac{\Delta y}{y} = \hat{\alpha} \cdot I\cos(\nu)/R = \hat{\alpha} \cdot \tan(2\mu).$$

This leads to the final equations:

$$\tan(2\mu) = \frac{1}{x_2} \cdot [\frac{y_2'}{y_2} - \frac{y_1'}{y_1}]. \tag{9}$$

and:

$$\hat{\alpha} = \frac{\frac{x'}{x} - \frac{y'}{y}}{\tan(2\mu)}\ ; \quad \tan(\nu) = \frac{\frac{\Delta y_1}{y_1}}{\tan(2\mu)}\ ; \quad R = \frac{I}{\sqrt{\tan^2(2\mu) + \frac{\Delta y_1}{y_1}^2}}.$$

The ratio $\frac{y'}{y}$ near the vertical meridian is relatively reliable and easy to obtain. However, a point with $\hat{\alpha} \approx 0$ does not necessarily exist, in which case we can:

1. Follow Mayhew & Longuet-Higgins (1982) and neglect the term $\hat{\alpha}\tan(\nu)$, but not the term $\tan(\nu)$.

$$\tan(2\mu) = \frac{1}{x} \cdot [\frac{y'}{y} - \frac{y_1'}{y_1}]. \tag{9'}$$

If we neglect tan(ν), for consistency, we get:

$$\tan(2\mu) = \frac{1}{x} \cdot \frac{\Delta y'}{y}. \qquad (9'')$$

2. Solve the initial scheme without such a point. Given a vertical-meridian point, there remains a fairly simple equation to solve. This would be:

$$\begin{aligned} \tan(2\mu) - \hat{\alpha} \cdot \frac{\Delta y_1'}{y_1} &= \frac{1}{x} \cdot \left(\frac{y'}{y} - \frac{y_1'}{y_1}\right) \\ \tan(2\mu) \cdot \hat{\alpha} &= \frac{x'}{x} - \frac{y'}{y}, \end{aligned} \qquad (10)$$

which reduces, after substituting tan(2μ) from the second equation in the first equation, to a second-degree polynomial in $\hat{\alpha}$.

A different numerical approach would be to use, for example, three images taken while moving on the base plane. Denote by (x_0, y_0), (x_1, y_1) and (x_2, y_2) the coordinates of the conjugate projections of some point P on the three images. Denote by α_1, μ_1 and ν_1 the angle of tilt, half the angle of convergence, and the angle of gaze respectively in the coordinate system defined as above by the first two images. Denote by α_2, μ_2 and ν_2 the same angles in the coordinate system defined by the last two images (see figure 2). For motion on a straight line we have:

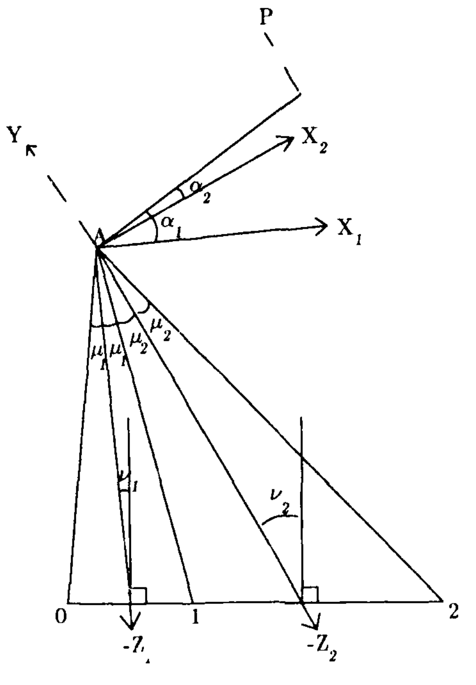


Figure 2 Three images from motion on the base plane

$$\tan\alpha_1 \cdot \tan\mu_1 = \frac{\lambda(x_0, y_0, x_1, y_1) - 1}{\lambda(x_0, y_0, x_1, y_1) + 1}$$

$$\tan\nu_1 \cdot \tan\mu_1 = \frac{1 - \frac{y_1(x=0)}{y_0(x=0)}}{1 + \frac{y_1(x=0)}{y_0(x=0)}}$$

$$\tan\alpha_2 \cdot \tan\mu_2 = \frac{\lambda(x_1, y_1, x_2, y_2) - 1}{\lambda(x_1, y_1, x_2, y_2) + 1}$$

$$\tan\nu_2 \cdot \tan\mu_2 = \frac{1 - \frac{y_2(x=0)}{y_1(x=0)}}{1 + \frac{y_2(x=0)}{y_1(x=0)}}$$

$$\alpha_1 - \alpha_2 = \mu_1 + \mu_2$$

$$\nu_2 - \nu_1 = \mu_1 + \mu_2,$$

where $\frac{y_j(x=0)}{y_i(x=0)}$ is the Y-axis ratio on the vertical meridian ($x = 0$) between conjugate points in images i and j.

Thus we have six nonlinear equations with six unknowns. For small μ_i's we have approximately a linear problem, where the solution is a null-vector of the approximating matrix. We will obtain a very similar set of equations if we take two points in the three images and ignore the equations involving the angle of gaze ν. In this case the motion in the base plane does not have to be in a straight line.

ERROR ANALYSIS

First, from the definition of λ and \ it follows that the base plane itself is singular in the sense that these orders are not defined for points on it. The same problem exists in the analysis of normals to surfaces of objects. One can, however, estimate the orders and normals by substituting $\frac{y_r}{y_l}$ of a matched point far from the base plane. More specifically, for $P = (x, y, z)$ we have

$$\begin{aligned} \frac{y_r}{y_l} &= \frac{d_l}{d_r} + \frac{z}{d_r}\frac{2\sin\mu\tan\nu}{1+\tan\mu\tan\nu} + \frac{x}{d_r}\frac{2\sin\mu}{1+\tan\mu\tan\nu} + o(\frac{x}{d_r}, \frac{z}{d_r}) \\ &= 1 - \frac{2\tan\mu\tan\nu}{1+\tan\mu\tan\nu} + \frac{z}{d_r}\frac{2\sin\mu\tan\nu}{1+\tan\mu\tan\nu} + \\ &\quad \frac{x}{d_r}\frac{2\sin\mu}{1+\tan\mu\tan\nu} + o(\frac{x}{d_r}, \frac{z}{d_r}). \end{aligned}$$

Thus, if point P^j is used to approximate point P^i, the error will be:

$$\left(\frac{y_r}{y_l}\right)^i - \left(\frac{y_r}{y_l}\right)^j = \frac{2\sin\mu}{1+\tan\mu\tan\nu}\left[\frac{z^i - z^j}{d_r}\tan\nu + \frac{x^i - x^j}{d_r}\right].$$

The error is 0 if the aproximating point P^j lies exactly "above" P^i (differs only in the y-coordinate).

Moreover, one can use as an estimation $\frac{1-\tan\mu\tan\nu}{1+\tan\mu\tan\nu}$ (the first two terms), so that some (possibly extraretinal) estimation of μ (half the angle of convergence) and ν (the angle of gaze) will suffice to give a rough estimation to $\frac{y_r}{y_l}$ when no other source of information is available. Note that one can't take $\frac{y_r}{y_l} \approx 1$ when computing $x_r - \frac{y_r}{y_l}x_l$, as a first order approximation in μ, since $x_r - x_l$ is of the order of magnitude of μ also.

Second, let us consider the violation of the basic assumption, namely, that the X-axes themselves of both cameras are epipolar lines. This introduces an error δ_r and δ_l in the polar angle of a given point's projections on the right and left images respectively. Thus, the true orders should be:

$$\lambda = \frac{\cot(\vartheta_r + \delta_r)}{\cot(\vartheta_l + \delta_l)}$$

$$= \frac{\cot\vartheta_r}{\cot\vartheta_l} + \frac{\cot\vartheta_r}{\cos^2\vartheta_l}\cdot\delta_l - \frac{\tan\vartheta_l}{\sin^2\vartheta_r}\cdot\delta_r + o(\delta),$$

$$\chi^y = \cot(\vartheta_r + \delta_r) - \cot(\vartheta_l + \delta_l)$$

$$= (\cot\vartheta_r - \cot\vartheta_l) - \frac{1}{\sin^2\vartheta_r}\cdot\delta_r + \frac{1}{\sin^2\vartheta_l}\cdot\delta_l + o(\delta),$$

where $\delta_r, \delta_l \leq \delta$. The main conclusion from this is that the effect of axes misallignment is greater near the horizontal and vertical meridians, and possibly negligible further away. Also, this error affects less the expressions for qualitative shape (the normal to iso-x or iso-y surface contours), since they involve differences where this error is somewhat cancelled out for the two points.

SUMMARY

The goal of this work had been to obtain qualitative information from a stereo pair, with as few computations as possible and minimal dependence on cameras' and scene's parameters. We have shown that points in a stereo pair, once matched to each other, can be ordered according to two distinct order expressions: a tilt-related order λ, which is roughly a relative depth order when ordering points with only vertical displacement, and a depth-related order χ which is best to order points with only horizontal displacement. These orders are completely determined by image coordinates of conjugate points, no camera or scene parameters are needed (which need not be similar for both cameras). λ and some variation of χ (χ^y) depend only on the polar angles of the conjugate points in both images, a quantity which is preserved under projection to a spherical body (an eye) or a planar body (a camera). Moreover, given the polar angles of the images coordinates, some qualitative shape *information can be obtained: one* can follow changes in the curvature of a contour on the surface of any object in the field of view, with either x- or y-coordinate fixed. We demonstrated, by further analyzing the exact equations, that obtaining the quantitative information is much harder and less reliable than obtaining the qualitative one, e.g., orders like λ and χ. It usually involves some assumptions on the scene or extra-retinal information, plus a lot of computations. These computations tend to be less robust and sensitive to noise and errors.

APPENDIX

Consider the base plane, which includes the X-axes of both cameras and both their optical-axes. This is illustrated in figure 3, where O is the focal-node of one camera, A- the fixation point, A'- the projection of A on the image plane or the origin of the camera coordinate system, B- the projection of a given point in space (C) on the base plane, B'- the projection of B on the camera X-axis, and C'- the projection of the point C on the image plane. In the base plane, we add the point D which is the projection of B on the optical axis $\overline{AA'}$. Let φ denote the angle $\angle BAO$. Let (x,y) denote the coordinates of the projection of point C on the image, so that $x = \overline{A'B'}$ and $y = \overline{B'C'}$. Let d denote the distance of the fixation point to the eye, so that $d = \overline{AO}$, and h denote the focal length of the camera, so that $h = \overline{A'O}$. Using similar triangles, one can verify the following:

$$\frac{\overline{BC}}{y} = \frac{\overline{BO}}{\overline{B'O}} = \frac{\overline{DO}}{\overline{A'O}} = \frac{d - \overline{AB}\cos\varphi}{h}.$$

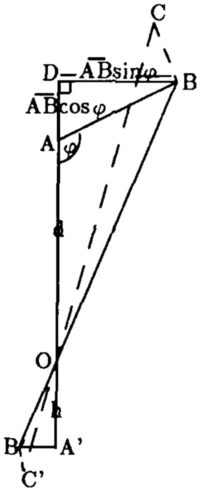


Figure 3. The base plane viewed from above, with one camera.

In other words,

$$y = \frac{h\overline{BC}}{d - \overline{AB}\cos\varphi}. \qquad (11)$$

Using the same arguments, we get

$$\frac{\overline{AB}\sin\varphi}{x} = \frac{\overline{DO}}{\overline{A'O}} \approx \frac{d - \overline{AB}\cos\varphi}{h}.$$

In other words,

$$x = \frac{h\overline{AB}\sin\varphi}{d - \overline{AB}\cos\varphi}.$$

Thus

$$\frac{x}{y} = \frac{\overline{AB}}{\overline{BC}}\cdot\sin\varphi.$$

Note that our assumption that the base plane intersects both cameras X-axes implies that the same geometry holds for both cameras in the sense that the segments $\overline{BC}$ and $\overline{AB}$ are identical in both cases. (A and C are the same points, and B is identical since C is projected onto the same plane.) Let us add indices for the variables of the left and right cameras, l and r respectively. Then

$$\frac{x_r}{y_r} = \frac{\overline{AB}}{\overline{BC}}\cdot\sin\varphi_r$$
$$\frac{x_l}{y_l} = \frac{\overline{AB}}{\overline{BC}}\cdot\sin\varphi_l. \qquad (12)$$

Finally

$$\frac{x_r}{y_r} / \frac{x_l}{y_l} = \frac{\sin\varphi_r}{\sin\varphi_l}. \qquad (13)$$

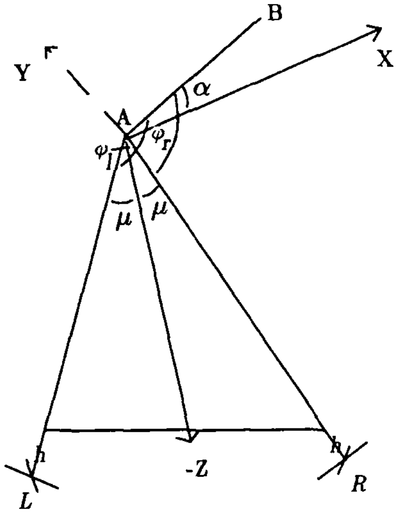


Figure 4. Angles of tilt on the base plane as viewed from above.

Figure 4 illustrates the geometry on the base plane with both cameras, and the coordinate system used in the text. Recall that 2μ denotes the angle of convergence of the cameras in the base plane, and that the Z-axis is defined as the angle-bisector of this angle. Also, α was defined as $\arctan(\frac{z}{x})$, so that $\alpha = \frac{\varphi_r+\varphi_l}{2} - 90^o$. Those definitions of μ and α imply the following:

$$\varphi_r = \alpha + 90^o - \mu$$
$$\varphi_l = \alpha + 90^o + \mu.$$

Thus

$$\frac{\sin\varphi_r}{\sin\varphi_l} = \frac{\cos\alpha\cos\mu + \sin\alpha\sin\mu}{\cos\alpha\cos\mu - \sin\alpha\sin\mu} = \frac{1+\tan\alpha\tan\mu}{1-\tan\alpha\tan\mu}. \qquad (14)$$

We have defined $\lambda = \frac{\cot\vartheta_r}{\cot\vartheta_l} = \frac{x_r}{y_r}/\frac{x_l}{y_l}$. From (13) and (14) we get

$$\lambda = \frac{1+\tan\alpha\tan\mu}{1-\tan\alpha\tan\mu} \implies \tan\alpha\tan\mu = \frac{\lambda-1}{\lambda+1},$$

which is equivalent to equation (1).

Now, if $C = (x, y, z)$ in the world coordinate system we have defined above, then

$$\begin{aligned}\cot\vartheta_r - \cot\vartheta_l = \frac{x_r}{y_r} - \frac{x_l}{y_l} &= \frac{\overline{AB}}{\overline{BC}}(\sin\varphi_r - \sin\varphi_l)\\ &= \frac{\overline{AB}}{\overline{BC}}2\sin\alpha\sin\mu\\ &= 2\sin\mu\frac{z}{y}\ .\end{aligned}$$

Since by definition $\tan\beta = \frac{z}{y}$, we immediately get equation (2).

Let us develop the expression for the image coordinate y above, considering the right image with no loss of generality. We then have from (11),

$$\begin{aligned}y_r &= \frac{h\overline{BC}}{d_r - \overline{AB}\cos\varphi_r}\\ &= \frac{hy}{d_r - x\sin\mu + z\cos\mu}.\end{aligned}$$

From figure 1, in which the angle of gaze ν is defined, it follows that $d_r = \frac{I\cos(\mu-\nu)}{\sin 2\mu}$, so that

$$y = (\frac{I\cos(\mu-\nu)}{\sin 2\mu} - x\sin\mu + z\cos\mu)\cdot\frac{y_r}{h}. \qquad (6)$$

Applying the same argument to the y coordinate of the left image, we get

$$y = (\frac{I\cos(\mu+\nu)}{\sin 2\mu} + x\sin\mu + z\cos\mu)\cdot\frac{y_l}{h}. \qquad (6')$$

ACKNOWLEDGEMENT: I thank Tomaso Poggio for his direction and support before and during the preparation of this manuscript.

REFERENCES

Foley, J. M., Primary distance perception, in: *Handbook of sensory physiology*, 1977.

Foley, J.M., and W. Richards, Effects of voluntary eye movement and convergence on the binocular appreciation of depth, *Perception & psychophysics* V-11(6), 423-427, 1972.

Horn B. K. P., "Robot Vision", MIT Press & McGraw-Hill, 1986.

Longuet-Higgins H. C., A computer algorithm for reconstructing a scene from two projections, *Nature* 293, 133-135, 1981.

Longuet-Higgins H. C., The role of vertical dimension in stereoscopic vision, *Perception* V-11, 377-386, 1982.

Longuet-Higgins H. C., and K. Prazdny, The interpretation of a moving retinal image, *Proc. R. Soc. Lond. B* 208, 385-397, 1980.

Mayhew, J. E. W., and H. C. Longuet-Higgins, A computational model of binocular depth perception, *Nature* V-297 (3), 376-378, 1982.

Prazdny, K., Stereoscopic matching, eye position, and absolute depth, *Flair Draft*, 1981.

Weinshall, D., Qualitative depth and shape from stereo, in agreement with psychophysical evidence, MIT-AI Memo 1007, 1987.

The Integration of Information from Stereo and Multiple Shape-From-Texture Cues

Mark L. Moerdler and Terrance E. Boult

Department of Computer Science
Columbia University, New York, N.Y. 10027

Abstract

In numerous computer vision applications, there is both the need for and the ability to access multiple types of information about the three dimensional aspects of objects or surfaces. When this information comes from different sources the combination becomes non-trivial.

This paper describes an approach that integrates multiple visual sensing methodologies yielding three dimensional information. The current system integrates feature based stereo algorithms with various shape-from-texture algorithms (multi-view shape-from-texture and shape-from-motion modules expected to be incorporated in the future). Unlike most systems for multi-sensor integration, that fuse all the information at one conceptual level, e.g., the surface level, the system under development uses two levels of data fusion, intra-process integration and inter-process integration. The paper discusses intra-process integration techniques for feature-based stereo and shape-from-texture algorithms. It also discusses an inter-process integration technique based on smooth models of surfaces. Examples are presented using camera acquired images.

1 Introduction

This paper discusses research into the integration of two different but highly compatible modalities: Shape-from multiple feature based stereo algorithms which use different types of features and multiple shape-from-texture algorithms which utilizes different types of textural cues. These modalities are being considered because they are, in general, applicable to similar regions of an images. This allows experimentation with both corroborating information, e.g., stereo images of textured three dimensional surfaces, and conflicting information (not reported here), e.g., stereo images of a two dimensional image of a textured three dimensional surface.

While solving the general fusion problem is beyond the current abilities of AI research, there has been progress in restricted contexts; these including fusion of stereo and tactile data [Allen 85; Allen 86], pointwise fusion of data [Henderson and Fai 83], and fusion of various information about an intensity image for the purpose of segmentation [Belknap et. al. 85; Kohler 84; Mckeown et. al. 84]. There has also been work on the regularization as a means for fusion [Blake and Zisserman 86; Medioni and Yasumoto 85; Poggio and Torre 84].

All of the above mentioned systems fuse their information at one conceptual level, e.g., the image or surface level. In contrast, the system under development uses two levels of data fusion, intra-process integration and inter-process integration. The former consists of fusion of information generated by all shape-from-X approaches with certain predetermined similarities, e.g., feature based stereo algorithms with different features. The latter type of integration is the fusion of the information resulting from each of the intra-process integration phases, with any *a priori* knowledge, e.g., smoothness assumptions or model assumptions. However, to allow for some amount of top-down processing and the future addition of world based constraints communication is performed, between each process, through a global blackboard.

The remainder of this paper is divided into sections on background and motivation, texture algorithms and related intra-process integration, stereo algorithms and related intra-process integration, and inter-process integration. Following the description of the current system, the results of limited experimental testing, using camera images, is presented.

2 Motivation and Background

As research in vision has progressed, it has been realized that the information available from a single "shape-from" algorithm would not be sufficient to solve the general vision problem. Prior vision research has yielded different "modalities" of information including: numerous approaches to shape-from-texture [Kender 80; Witkin 80], binocular stereo [Marr and Poggio 79; Eastman and Waxman 85; Hoff and Ahuja 85], shape-from shading [Horn 70; Lee 85], and shape-from-motion [Anadan and Weiss 85; Prazdny 79]. Each of these sources of shape information have different domains of applicability, different computational complexity, and different error characteristics. For any given module, there exists numerous images (or regions there of) for which the module would not correctly predict surface shape. Some of the sources are complementary, e.g. shape-from-shading will apply generally only in those regions where shape-from-texture will fail. Other modules can act in either a competitive or synergistic fashion, e.g. binocular stereo and shape-from-texture will generally apply in the same regions of an image, and may compete for dominance if their outputs differ, or can mutually reinforce a consistent interpretation.

[1] This work was supported in part by ARPA grant #N00039-84-C-0165.

Robustness could therefore be gained by combining different shape-from-X methods within a single unified system. In designing such a system there exist two basic problems:

1. Different modalities generate information constraints based on different scale assumptions. If the shape-from methods utilize the same scale and type of information then all of the constraints that relate to the same image area can be integrated to generate a single date element for that image area, e.g. fusing all of shape-from-texture methods. If the methods utilize different scale and/or different date type then a common level must be chosen.
2. Generating a single interpretations from the large numbers of cooperating and/or conflicting information sources is predicated on choosing a "reasonable" confidence weighting for each constraint. The confidence weighting should be comprised of two basic components:
    - An intra-method weighting that states how closely the image data fits the shape-from method's underlying assumptions.
    - An inter-method weighting which assures that no single method dominates because it generates substantially more constraints. Instead, a method should dominate because its constraints define a single datum of information whose confidence is higher than another method(s). Model driven expectations can also be included in this weight if there exists a predeliction for one method over another.

3 Integrating Methodologies

The two-level fusion methodology decouples data acquisition and its related assumptions from the final results of data fusion and surface generation. The lower level intra-process integration techniques derives orientation information based on the underlying level of abstraction. The explicit and implicit assumptions of the underlying intra-shape-from-X methods are used to weigh the confidence of each orientation constraint in relationship to constraints generated by related intra-shape-from-X methods (e.g. constraints from shape-from-uniform-texel-spacing with constraints from shape-from-uniform-texel-size). A brief presentations is given of two such intra-process integration techniques.

Since the final result of the data fusion is assumed to be objects with smooth surfaces, the inter-process integration depends on our model of surfaces. The current system employs a regularization based surface reconstruction technique for this task. This approach allows the system to independently weight each piece of information from the intra-process integration phases. Part of this weighting is a global factor determining which of the modalities has higher priority.

The interaction among the various computational modules as well as the integration modules, is accomplished with a blackboard organization. This scheme allows bidirectional flows of information and provides a means for easy detection of when the information necessary for the execution of each modules is present in the system. This portion of the system will not be considered further in this paper.

A two level integration approach has two additional advantages. The first is computational in nature, and derives from the fact that it is easier to heuristically combine data from similar sources, as in the intra-process integration phase. Then, when the system must integrate information from markedly different sources, that information should already be of higher quality than the initial raw data. This separation of duties also aids in maintaining system modularity, and minimizes global memory requirements.

The second reason for desiring a multi-level integration scheme follows from studies of human vision. Considerable research exists on the human perception of three-dimensional surfaces [julesz-61; bulthoff-mallot-87]. Many of these works have used selective stimuli, e.g., random-dot stereograms, to study phenomenological aspects of depth perception from various sources. From these studies one can draw inferences about the integration of information from "modules" using this information. Other works have studied the "ordering" of operations in the human visual system and the interaction of various information sources for depth perception. Of particular relevance to the question of multi-level integration is the work of [bulthoff-mallot-87], which examined the interaction, both rivalarous and mutually supportive, among various information sources.

4 Texture processes and texture intra-process integration

This section discusses our approach to the problem of deriving orientation information from multiple independent textual cues. The method consists of two major phases: the generation of constraints on the orientation of *texel patches*[1], and intra-process integration where the orientation constraints, for each patch, are fused into a "most likely" orientation. The robustness of this approach has already been demonstrated elsewhere, see [Moerdler and Kender 87a; Moerdler and Kender 87b; Moerdler 88].

Currently the shape-from-texture methods used are: shape-from-uniform-texel-spacing [Kender 83], and shape-from-uniform-texel-size [Ohta et. al. 81]. These two methods generate orientation constraints for different overlapping classes of textures.

4.1 Background

Current methods to derive shape-from-texture are based on measuring a distortion that occurs when a textured surface is viewed under perspective, assuming of course that natural texture neither mimics nor cancels projective effects. The perspective distortion results in some aspect of the texture being deformed when the scene is imaged. In order to simplify the recovery of the orientation parameters from this distortion, researchers have imposed limitations on the applicable class of textured surfaces. Some of the limiting assumptions include uniform texel spacing [Kender 80; Kender 83; Moerdler and Kender 85], uniform texel size [Ikeuchi 80; Ohta et. al. 81],

[1] A *texel patch* is a 2-D description of a sub-image that contains one or more textural elements. The number of elements that compose a patch is dependent on the shape-from-texture algorithm.

uniform texel density [Aloimonos 86], and texel isotropy [Witkin 80; Dunn 84]. These are strong limitations causing methods based on them to be applicable to only a limited range of real images.

4.2 Design Methodology

The generation of orientation constraints from perspective distortion is performed using one or more image texels. The orientation constraints can be considered as local, defining the orientation of individual surface patches called *texel patches*. Texel patches are defined by how each method utilizes the texels. Some methods, e.g., uniform texel size, use a measured change between two texels; in this case the texels patches are the texels themselves. Other methods, e.g., uniform texel density, use a change between two areas of the image. In the latter case the texel patches are predefined areas of the image. For the texture modules, intra-process fusion is carried out at the texel patch level.

This differs from integration at the surface level which has been attempted elsewhere (e.g., [Ikeuchi 80] and [Aloimonos 86]) and use constraint propagation and relaxation to derive a single orientation per surface patch.[2]

The process of fusing orientation constraints and generating surfaces can be broken down into the following three phases: (1) the creation of texel patches, (2) calculation of (multiple) orientation constraints for each texel patch, and (3) the unification of the orientation constraints per texel patch into a "most likely" orientation. Each of the remaining portions of this subsection describes one of these phases.

4.2.1 Texel patch definition

There has been considerable work in computer vision on the automatic recognition of textural patches. While accurate and consistent texel patch recovery would greatly simplify the integration process, we feel that such data is unavailable at the present time. Instead, we have chosen a simplistic patch definition obtained by first processing the image with assorted filters and then thresholding the image to define patches.[3] We acknowledge that better texture discrimination algorithms exist, but this was not the focus of our research. In the work described in this paper (and described in greater depth in [Moerdler 88]) we have filtered the image by local averaging of the gray levels. We have also experimented with edge detection and edge orientation filters (with and without post filtering smoothing), although those filters are not used on the examples herein.

4.2.2 Surface Patch and Orientation Constraint Generation

The first phase of the system consists of several shape-from-texture components generating augmented texels. Each augmented texel consisting of a texel patch, orientation constraints for the texel patch, and an confidence weighting per constraint. The orientation constraints are stored in the augmented texel as vanishing points which are mathematically equivalent to a class of other orientation notations (e.g. pan and tilt constraints) [Shafer, Kanade, and Kender 83]. Moreover, they are simple to generate and compact to store.

The confidence weighting is defined separately for each shape-from method and is based upon the intrinsic error of the method. For example, shape-from-uniform-texel-spacing's confidence weighting is a function of the total distance between the texel patches used to generate that constraint. The confidence measure decreases as the distance between the texels increase because once the inter-texel distance grows too large the local surface is no longer approximated by a plane and the orientation error grows. This further acts to make the constraints group locally rather than globally which is valid since texels that are part of the same surface are normally located close together.

The current system contains two shape-from-texture methods: shape-from-uniform-texel-spacing [Moerdler and Kender 85] based on the assumption that the texels can be of arbitrary shape but are equally spaced , and shape-from-uniform-texel-size [Ohta et. al. 81] based on the unrelated criteria that the spacing between texels can be arbitrary and the size of all of the texels is equivalent but unknown. Each of the methods is based on a different textural characteristic that allows the generation of orientation constraints and also limits the applicability of the approach. Future plans call for the inclusion of shape-from-convergence [Kender 80], and shape-from-ellipticity of circular textures.

In shape-from-uniform-texel-size two texels are used T_1 and T_2 whose sizes are S_1 and S_2 respectively. If the distance from the center of mass of texel T_1 to texel T_2 (see figure 1) is defined as D then the distance from the center of texel T_2 to a point on the vanishing line can be written as :

$$\frac{F_1}{F_2} = \frac{S_1^{1/3}}{S_2^{1/3}}$$

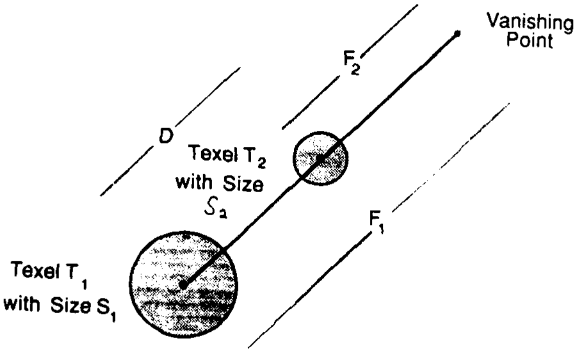


Figure 1: The calculation of shape-from-uniform-texel-size

In shape-from-uniform-texel-spacing the calculations are similar. Given any two texels T_1 and T_2 (see figure 2) whose inter-texel distance is defined as D, if the distance from T_1 to a mid-texel T_3 is equal to L and the distance from T_2 to the same mid-texel T_3 is equal to R, the distance from texel T_1 to a vanishing point is given exactly by :

[2] These approaches were further limited by the use of only a single shape-from-texture method.

[3] An analysis of this type of technique for texture discrimination can be found in [Davis et.al. 84].

$$X = \frac{L \times P_1 - R \times P_2}{L - R}$$

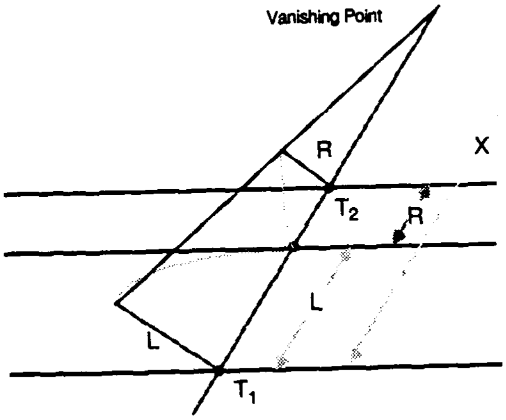


Figure 2: A geometrical representation of back-projecting.

4.3 Intra-Process Integration for Textures: Generation of Most Likely Orientation

Once the orientation constraints have been generated for each augmented texel, the next step consists of unifying the constraints into one orientation per augmented texel. A simple and computationally feasible method of integrating the constraints generated by the intra-shape-from-texel is to use a Gaussian Sphere which maps the orientation constraints to points on the sphere [Shafer, Kanade, and Kender 83]. A single vanishing point circumscribes a great circle on the Gaussian Sphere; two different constraints generate two great circles that overlap at two points uniquely defining the orientation of both the visible and invisible sides of the surface patch.

The Gaussian sphere is approximated within the module by the hierarchical tesselated Gaussian Sphere based on triangular shaped faces called trixels [Fekete and Davis 84]. The top level of the hierarchy is the twenty face icosahedron. At each level, other than the lowest level of the hierarchy, each trixel has four children which more closely approximate the curvature of the spherical surface than their parent. This hierarchical methodology allows the user to specify the accuracy to which the orientation can be calculated by defining the number of levels of tesselation that are created.

The intra-texture-process integration phase generates the "most likely" orientation for each texel patch by accumulating the evidence from all the orientation constraints (generated in phase one) for the patch. For each constraint, it initially visits the twenty top level trixels, determines whether the great circle falls on the trixel and if the result is positive, visits the children. At each lowest level trixel through which the great circle travels, the likelihood value of the trixel is incremented by the constraint's weight. The hierarchical nature of this approach limits the number of trixels that need to be visited. Once all of the constraints for a texel patch have been considered, a peak finding program smears the likelihood values at the lowest level trixels. The "most likely" orientation is defined to be the trixel with the largest smeared value.

This method does not assure that under all circumstances a single "most likely" orientation will be derived. When more than one "most likely" orientation is derived for a patch the module performs a Waltz type filtering. It computes the "most likely" orientation for the remaining augmented texels and then re-analyzes the orientation constraints for each texel that does not have a single "most likely" orientation. For each unsolved texel patch the module considers all of the patch's constraints and removes any constraints that do not correctly define the "most likely" orientation of another texel patch. Once this constraint pruning has occurred the module recomputes the "most likely" orientation for the patch. This secondary analysis does not assure a single "most likely" orientation either, but it does aid in simplifying and deriving a single "most likely" orientation for the largest number of surface patches.

5 Stereo processes and intra-process integration

The stereo-based processes of the system are based on matching features between the two images. The system uses multiple feature definitions to insure both good localization and noise resistance. These feature are then classified as to amount of ambiguity. The system starts with the least ambiguous matches and reconstructs a disparity surface. Intra-stereo-integration is accomplished through a regularized reconstruction of the disparity field based on the assumption that the smooth surfaces in the world give rise to a smooth disparity surface. After all points are considered, the intra-process module adds its output to the blackboard. Currently this output is depth values at various points, especially along the "edges" of surfaces in the disparity field and at the locations of feature points.

5.1 Definition of the multiple features

A common problem in stereo systems is that the features are too sparse, have poor localization, or are sensitive to noise. Rather than attempt to define yet another feature for matching, the stereo module currently combines two different types of features. These are: (1) zero crossings of laplacian of gaussians of the images, which are subsequently thresholded (based on magnitude of crossing) and matched along approximately epi-polar lines using orientation and sign as a filters), and (2) centroids of texels defined in the shape-from-texture algorithm (with some of the other texel features used to insure only valid matches). The first of these features, zero crossing, provide a large number of features for the matching algorithm, unfortunately the localization of these features are not highly accurate. The second set of features, texel centroids, are not very dense, however, they provide very accurate localization of the feature.

In the future we will be adding features derived from area based correlations, interest operators (e.g. [Moravec 79]) and a thresholded sobel operator.

5.2 Intra-stereo-integration module

The integration of the various features is accomplished by a multi-pass matching algorithm, where the quality of localization/ambiguity effects the order in which points are considered, and previously matched points effect the disambiguation of other points. The matching algorithm used is described in detail elsewhere, [Boult and Chen 87], only a brief description is presented here. The basic assumption underlying the matching algorithm is that the disparity surface should be smooth. In vision research, there have been many model based matching algorithms proposed with different "smoothly" varying disparities, [Marr and Poggio 79; Mayhew and Frisby 81; Eastman and Waxman 85]. The smooth disparity fields used for this system are based on generalized two dimensional smoothing splines, see [Boult 86]. The smoothness criterion is similar to one used in smooth surface reconstruction, see [Blake and Zisserman 86; Boult 86; Grimson 79; Terzopoulos 84].

The system starts with the feature points which have "unique matches" and good localization (i.e., at the current time it begins with the centroids of the "texel"). In all neighborhoods without these features, lower quality (in terms of localization) features with "unique" matches are added, however they are given a lower confidence value. Thus when the smooth surface is fitted to the disparity data, the disparity values generated by lower quality features will not be as closely approximated.

After all the "unique" matches have been used, the module reconstructs a disparity surface. This reconstruction is based on the assumption that the disparity surface should give rise to a surface with smooth depth changes. Using this disparity surface, the module disambiguates other matches by choosing the potential match which comes closest to the smooth surface. The distance between the disparity predicted by the "best" match and the smoothed disparity surface affects the confidence of the match, which in turns affects the way the disparity surface approximates that match. The disambiguation takes place in multiple passes each of which incorporates features that are increasingly ambiguous.

After all points are considered, the intra-process module adds its output to the blackboard. Currently, this output is depth values at various points, especially along the "edges" of surfaces in the disparity field, and depends on the calibration of the imaging system. Future plans call for the module to output surface orientation information in the place of depth data, thereby eliminating the need for calibration.

6 Inter-process integration and surface reconstruction

This section describes the inter-process integration phase of the system. This phase of the fusion process is predicated on the assumption that the world is comprised of piecewise smooth surfaces/objects. Therefore, the inter-process integration should depend on the assumed smoothness model(s) for surfaces, not on the data acquisition techniques. There are two main aspects of the inter-process integration, basic surface building, and the weighting of various modules.

6.1 Basic Surface Reconstruction Technique

Inasmuch as they can be expressed in terms of inverse optics, many problems in computer vision, including surface reconstruction from sparse 3D information, are ill-posed. One way of reformulating these ill-posed problems is through a well known technique called regularization.[4] Let us precisely define the problem at hand:

Let F_I, the space of allowed surfaces, be a Hilbert or semi-Hilbert space. Let $\|\bullet\|F_I$ be a norm (or semi-norm if F_I is semi-Hilbert) measuring the "unreasonableness" of a surface f. Let $N_n(f) = [L_1(f), ..., L_n(f)]$ be the given information. Then the visual surface reconstruction problem is to find $f^* \in F_I$ such that

$$\vartheta(f) = \min_{\bar{f} \in \mathcal{F}_1} \vartheta(\bar{f}) \text{ where } \vartheta(\cdot) \text{ is defined as:}$$

$$\vartheta(\bar{f}) \overset{\text{def}}{=} \lambda \cdot \|\bar{f}\|_{\mathcal{F}_1} + \sum_{i=1}^{n} \left|\delta_i \times L_i(\bar{f}) - L_i(f)\right|^2$$

The norm (semi-norm) $\|\bullet\|_{f_I}$ is generally refereed to as the *stabilizing functional* of the regularization. The class of functions F_I is an often overlooked, but immensely important, part of the regularization. One cannot indiscriminately choose how to regularize a problem. As pointed out in [page 315] [Poggio et.al. 85][5],

> "... standard regularization methods have to be applied after a careful analysis of the ill-posed nature of the problem. The choice of the $\|\bullet\|_{f_I}$ of the stabilizing functional $\|\bullet\|_{f_I}$ and of the functional spaces involved is dictated by both mathematical properties and by physical plausibility. They determine whether the precise conditions for a correct regularization hold for any specific case."

The use of regularization for reconstruction of smooth surfaces in vision was first proposed in [Grimson 79]. In that pioneering work Grimson discussed the choice of the most appropriate stabilizing functional (though in different terminology). However, his decision was partially based on an assumed relationship between the stabilizing functional and the zero-crossings of the intensity images which lead to the stereo depth data. With respect to visual surface reconstructions, many researchers (e.g., [Terzopoulos 84; Hoff and Ahuja 85; Poggio et.al. 85; Lee 85]) seem to have accepted the choice of norm, stabilizing functional and associated functional spaces that were initially proposed in [Grimson 79].[6]

However, the authors feel that this class is "too smooth" and thus have adopted the assumption that world surfaces can be piecewise modeled as surfaces belonging to the class of $D^{-2}H^{-.5}$ with the second Sobolev semi-norm, which can intuitively be described[7] as having only 1.5 derivatives in L^2. In work reported elsewhere, [Boult 87], this assumption was shown to be at least as reasonable as assuming surfaces in $D^{-2}L^2$. The reconstruction of the surfaces might be accomplished with discrete regularization techniques. However, the approach taken herein is based on generalized smoothing spline functions, see [Boult 86] and the references therein, and is more efficient for sparse data, see [Boult 85].

[4] There exist numerous ways to calculate such surfaces, see Chapter 9 [Boult 86] for a critical comparison of 4 methods.

[5] The mathematical notation in the quote has been modified to match that used in this paper.

[6] It is not clear if this acceptance is simply because the method gave reasonable results and there was a lack of alternatives, or because the choice is actually the most appropriate.

[7] For a precise definition see [Boult 86; Boult 87; Meinguet 79].

The system allows for each data point to be individually weighted in terms of its contribution to the allowed fitting error (the terms δ_i in the above definition). The choice of these weights is influenced by two things, the confidence passed for the point from the intra-process integration processes, and the weighting assigned to the module as a whole.

6.2 Weighting the outputs of the various intra-process integration modules.

The above fusion scheme requires that each data point be given a weight. The correct selection of these weights is difficult. The study of these weighting will be of paramount importance in our future research. Currently, the system builds three surfaces:

1. One surface from the output of the intra-texture-integration module using the weighting supplied by that module,
2. One surface from the output of the intra-stereo-integration module using the weighting supplied by that module, and
3. One surface combining all data. For the combination, the weights are divided by the number of data-points output by a module. This provides some means for the texture data to have an effect on the surface. Otherwise the stereo data (with 500-5000 points) would totally dominate the shape information from texture (which only provides ~10-50 data points).

While it would be nice for the system to choose which of these surface is the "correct" one, this is not possible. When the information is conflicting, the "correct" precept is subjective, and can often be changed by will in humans. However, when the surfaces agree, the system should be able to (but currently cannot) take note, and remove the redundant representations.

7 Experimentation

The system described in this paper is still under development and has only been subject to limited experimental testing. Presented in this section are two examples of the system working on camera images. The results are presented in figures 3 to 12. One example, the curved roll of paper (see figure 3) demonstrates a surface where stereo dominates, but is significantly aided by the texture information. In the other example (see figure 8, texture was more successful because the scan line coherence assumed in stereo was violated by a slight rotation of the scene. The stereo module was thus slightly off on each "texel" and produced a more bumpy surface (not shown).

8 Conclusions and future plans

This paper describes an ongoing research project that integrates two modalities: shape-from-texture and information from stereo. The system makes use of a two level integration scheme and while the experimental analysis is not complete, initial results show this multi-level integration to be both efficient and effective.

Figure 3: Left and right input images for example 1: An artificially textured roll of paper

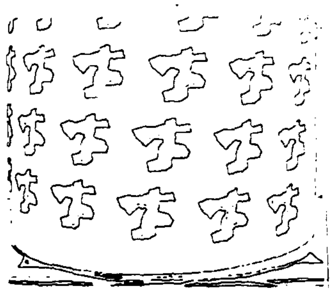


Figure 4: Laplacian of Gaussian of left image for Stereo

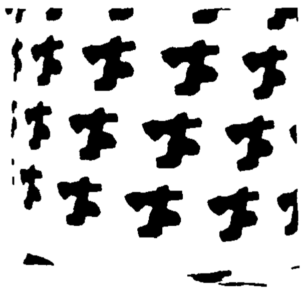


Figure 5: Texels identified in left image

Future work will include the addition of modules for shape-from-multi-view-texture, shape-from-shading, and possibly shape-from-motion. In addition, the number of processes in both the existing stereo and texture modules will be expanded.

An important task awaiting these researchers is the study of the weighting of the outputs of the various modules, and some attempt at development of non-ad hoc criterion.

9 Acknowledgments

The authors are grateful to John Kender for his many discussions on this subject. Also thanks to L.H. Chen without whom the stereo modules would not have been finished this year.

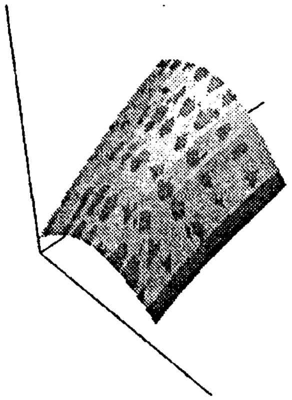


Figure 6: Reconstruction from just stereo data

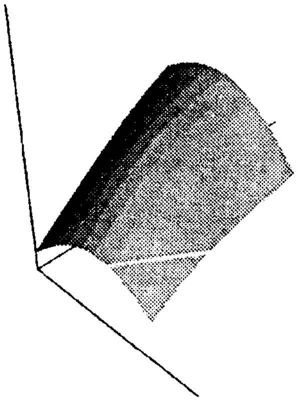


Figure 7: Reconstruction from combined stereo and texture data

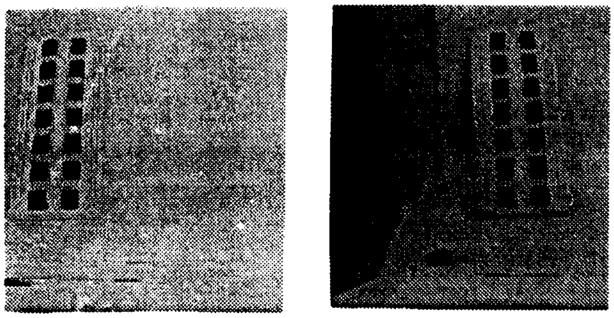


Figure 8: Left and right input images for example 2: "Circuit Breadboard"

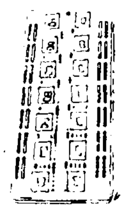


Figure 9: Laplacian of Gaussian of left image for Stereo

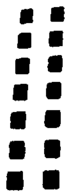


Figure 10: Texels identified in left image

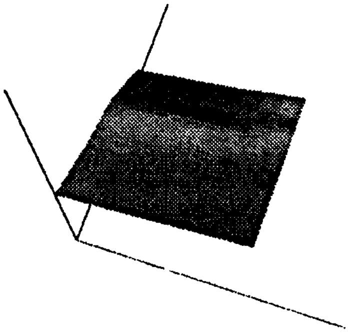


Figure 11: Reconstruction from just texture data

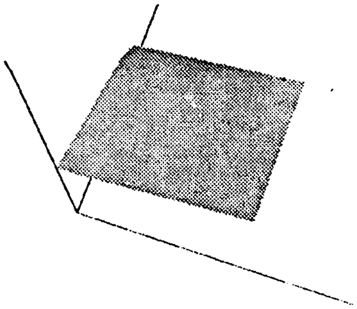


Figure 12: Reconstruction from combined stereo and texture data

References

[Allen 85] Peter K. Allen. *Object Recognition Using Vision and Touch.* PhD thesis, University of Pennsylvania, Department of Computer Science, 1985.

[Allen 86] Peter Allen. Sensing and describing 3-D structure. *Proc. IEEE Conference on Robotics and Automation* :126-131, 1986.

[Aloimonos 86] John Aloimonos. Detection of Surface Orientation and Motion from Texture: 1. The Case of Planes. In *Proceedings of Computer Vision Pattern Recognition Conference.* Computer Vision Patttern Recognition, IEEE Computer Society, 1986.

[Anadan and Weiss 85] P. Anandan and R. Weiss. Introducing a Smoothness Constraint in a Matching Approach for the Computation of Displacement Fields. In *the Proceedings of the DARPA Image Understanding Workshop*, pages 186-195. DARPA, Scientific Applications International, Inc., 1985.

[Belknap et. al. 85] R. Belknap, E. Riseman, and A. Hanson. The Information Fusion Problem and Rule-based Hypotheses Applied to Complex Aggregation of Image Events. In *Proceedings of the DARPA Image Understanding Workshop*, pages 279-292. DARPA, Scientific Applications International, Inc., 1985.

[Blake and Zisserman 86] A. Blake and A. Zisserman. Invariant Surface Reconstruction using Weak Continuity Constraints. In *Computer Vision Pattern Recognition*, pages 62-68. IEEE, 1986.

[Boult 85] T. E. Boult. Visual Surface Interpolation: A Comparison of Two Methods. In *DARPA Image Understanding Workshop*, pages 446-478. DARPA, Scientific Applications Intl., Inc., 1985.

[Boult 86] Terrance E. Boult. *Information Based Complexity in Non-Linear Equations and Computer Vision.* PhD thesis, Department of Computer Science, Columbia University, 1986.

[Boult 87] T.E. Boult. What is regular in regularization? In *First International Conference on Computer Vision*, pages 457-462. IEEE, Computer Society Press, June, 1987.

[Boult and Chen 87] T.E. Boult and L.H. Chen. Analysis of two new stereo algorithms integrating surface reconstruction and matching. *submitted for publication* , 1987.

[Bulthoff and Mallot 87] H.H. Ulthoff and H.A. Mallot. Interaction of Different Modules in Depth Perception. In *the First Internation Conference on Computer Vision*, pages 295-305. IEEE, Computer Society Press, 1987.

[Davis et.al. 84] L. S. Davis and A. Rosenfeld and J.S. Weszka. *Region Extraction by Averaging and Thresholding.* Technical Report TR-311, F44620-72C-0062, University of Maryland Center For Automation Research, June, 1984.

[Dunn 84] Stanley M. Dunn, Larry S. Davis, and Hannu A. Hakalahti. *Experiments in Recovering Surface Orientation from Texture.* Technical Report CAR-TR-61, University of Maryland Center For Automation Research, May 1984.

[Eastman and Waxman 85] R.D. Eastman and A.M. Waxman. Disparity Functionals and Stereo Vision. In *the Proceedings of the DARPA Image Understanding Workshop*, pages 245-254. DARPA, Scientific Applications International, Inc., 1985.

[Fekete and Davis 84] G. Fekete and L. S. Davis. Property Spheres: A New Representation For 3-D Object Recognition. *Proceedings of the Workshop on Computer Vision Representation and Control* :192 - 201, 1984.

[Grimson 79] W. E. L. Grimson. *From Images to Surfaces: A Computational Study of the Human Visual System.* PhD thesis, MIT, 1979.

[Henderson and Fai 83] T. Henderson and W.S. Fai. *Pattern Recogntion in a Multi-Sensor Environment.* Technical Report 83-001, Univ. of Utah, 1983.

[Hoff and Ahuja 85] W. Hoff and N. Ahuja. Surfaces from Stereo. In *the Proceedings of the DARPA Image Understanding Workshop*, pages 98-106. DARPA, Scientific Applications International, Inc., 1985.

[Horn 70] B.K.P. Horn. Shape from Shading: a Method for Obtaining the Shape of a Smooth Opaque Object from One View. *Technical Report MAC-TR-79* , Project MAC, MIT, 1970.

[Ikeuchi 80] Katsushi Ikeuchi. Shape from Regular Patterns (an Example from Constraint Propagation in Vision). *Proceedings of the Internation Conference on Pattern Recognition* :1032-1039, 1980.

[Julesz 60] B. Julesz. Binocular Depth Perception of Computer Generated Patterns. *Bell Systems Technical Journal* 39:1125-1161, 1960.

[Kender 80] John R. Kender. *Shape from Texture.* PhD thesis, Carnegie Mellon University, 1980.

[Kender 83] J.R. Kender. Environmental Labelings in Low-Level Image Understanding. In *Proceedings of the Eighth International Joint Conference on Artificial Intelligence.* 1983.

[Kohler 84] R. R. Kohler . *Integrating Non-Semantic Knowledge into Image Segmentation Processes.* COINS Technical Report 04, U. Mass. at Amherst , March, 1984.

[Lee 85] D. Lee. *Contributions to Information-based Complexity, Image Understanding, and Logic Circuit Design.* PhD thesis, Columbia University, 1985.

[Marr and Poggio 79] D. Marr and T. Poggio. A Computational Theory of Human Stereo Vision. *Proceeding Royal Society of London.* B(204):301-328, 1979.

[Mayhew and Frisby 81] J.E.W. Mayhew and J. P. Frisby . Psychological and Computational Studies towards a Theory of Human Stereopsis. *Artificial Intelligence* 17:349-385, 1981.

[Mckeown et. al. 84] D. M. Mckeown, W.A. Harvey and J. McDermott . *Rule-based Interpretation of Aerial Imagery* . Technical Report, CMU, September, 1984.

[Medioni and Yasumoto 85] G. Medioni and Y. Yasumoto. Robust Estimation of 3-D Motion Parameters from a Sequence of Image Frames using Regularization. In *DARPA Image Understanding Workshop*, pages 117-128. DARPA, Scientific Applications Intl., Inc., 1985.

[Meinguet 79] J. Meinguet. Multivariate Interpolation at Arbitrary Points Made Simple. *J. of Applied Mathematics and Physics (ZAMP)* 30:292-304, 1979.

[Moerdler 88] Mark L. Moerdler. *Shape-From-Textures:A Paradigm for Fusing Middle Level Vision Cues.* PhD thesis, Columbia University, Department of Computer Science, In Process due 1988.

[Moerdler and Kender 85] Mark L. Moerdler and John R. Kender. *Surface Orientation and Segmentation from Perspective Views of Parallel-Line Textures.* Technical Report, Columbia University, 1985.

[Moerdler and Kender 87a] Mark L. Moerdler and John R. Kender. An Integrated System That Unifies Multiple Shape From Texture Algorithms. In *Proceedings of AAAI 87*, pages 723 to 727. AAAI, Morgan Kaufman Publishers, Inc., 1987.

[Moerdler and Kender 87b] Mark L. Moerdler and John R. Kender. An Approach to the Fusion of Multiple Shape From Texture Algorithms. In *Proceedings of the Workshop on Spatial Reasoning and Multi-Sensor Fusion*, pages 272 to 281. Morgan Kaufman Publishers, Inc., 1987.

[Moravec 79] H.P. Moravec. Visual Mapping by a Robot Rover. In *Proceedings of the Sixth International Joint Conference on Artificial Intelligence*, pages 598-600. 1979.

[Ohta et. al. 81] Yu-ichi Ohta, Kiyoshi Maenobu, and Toshiyuki Sakai. Obtaining Suface Orientation from Texels under Perspective Projection. In *Proceedings of the Seventh International Joint Conference on Artificial Intelligence.* IJCAI, IJCAI, 1981.

[Poggio and Torre 84] T. Poggio and V. Torre . *Ill-posed Problems and Regularization in Early Vision.* Technical Report lab memo 773, Massachusetts Institute Technology AI, 1984. Also appeared in the Proceedings of the DARPA Image Understanding Workshop.

[Poggio et.al. 85] T. Poggio and V. Torre and C. Koch. Computational vision and regularization theory. *Nature* 317(6035):314-319, 1985.

[Prazdny 79] K. Prazdny. *Egomotion and Relative Depth Map from Optical Flow.* PhD thesis, University of Essex, 1979.

[Shafer, Kanade, and Kender 83] Steven A. Shafer and Takeo Kanade and John R. Kender. Gradient Space under Orthography and Perspective. *Computer Vision, Graphics and Image Processing* (24):182-199, 1983.

[Terzopoulos 84] D. Terzopoulos. *Multiresolution Computation of Visible-Surface Representations.* PhD thesis, MIT, 1984.

[Witkin 80] Andrew P. Witkin. Recovering Surface Shape from Orientation and Texture. *Computer Vision.* North-Holland Publishing Company, 1980, pages 17-45.

STRUCTURAL CORRESPONDENCE IN STEREO VISION[*]

Hong Seh Lim[†]and Thomas O. Binford

Robotics Laboratory, Computer Science Department,
Stanford University, Stanford, CA 94305, U.S.A.

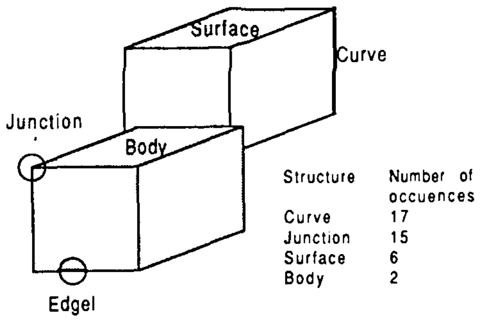
Abstract

This paper describes additional results on the work *[Lim 1987a] which appeared in last Image Understanding Workshop.* We design and implement a high-level stereo-vision system which achieves global correspondence between high-level structures of stereo pairs. This structural correspondence reduces matching complexity, and provides globally consistent matching results avoiding local mismatches. Recovered range data is segmented into surfaces and bodies.

1 Introduction

We perform stereo correspondence using five different types of structures: bodies, surfaces, curves, junctions, and edgels (Figure 1). These structures are segmented and grouped into a hierarchical structure based on inclusion (Figure 2 and Figure 3). The matching of complex structures at the top level is used to guide and constrain the matching of simpler structures at lower levels of the hierarchical structures (Figure 4). This is the first system to perform matching at the level of surfaces and bodies.

The system applies monocular interpretation in order to segment individual images before matching. Given a pair of stereo images (Figure 5), edgels are detected using the Nalwa edge operator [Nalwa 1986]. Detected edges are then fitted with straight lines or conics [Nalwa 1987] (Figure 6). We extend curves around junctions to locate missing edges and form new junctions (Figure 7). These junctions are classified accordingly. Surfaces are segmented from curves and are grouped into separate bodies (Figure 8 and Figure 9). Because of this segmentation effort, we are matching symbolic structures. The resulting three dimensional depth data is segmented and has symbolic meanings attached to the parts (Figure 10). Thus it is not necessary to perform further segmentation in three-dimensional space as other systems must.



Structure	Number of occuences
Curve	17
Junction	15
Surface	6
Body	2

Figure 1: *Higher-level Structures have Fewer Occurences*

Because of the hierarchical structure, the number of bodies at the highest level is two orders of magnitude less than the number of edgels at the lowest level. We show that the complexity of the matching is thus reduced by at least four orders of magnitude in time.

Section 2 explains in details the structures chosen for stereo correspondence. The structures include edgels, curves, junctions, surfaces, and bodies. They are segmented into a hierarchical structure by edge detection, curve fitting, curve extension, junction classification, and monocular interpretation. Some viewpoint sensitive and invariant properties of these structures are discussed.

Section 3 focuses on the matching strategy. The epipolar constraint is extended to surfaces and bodies. This is a major constraint used in this work. It is used for determining the matching of bodies and surfaces. A hierarchical constraint is introduced. We show that the matching complexity is reduced by four orders of magnitude through the hierarchical arrangement of

[*] This work was supported by the Defense Advanced Research Projects Agency under contract number N00039-84-C-0211

[†] Present Address: IBM Corporation, Los Angeles Scientific Center, 11601 Wilshire Boulevard, Los Angeles, CA 90025

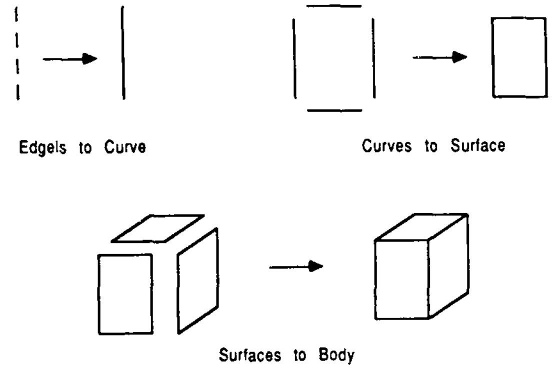


Figure 2: Grouping of Structures by Inclusion

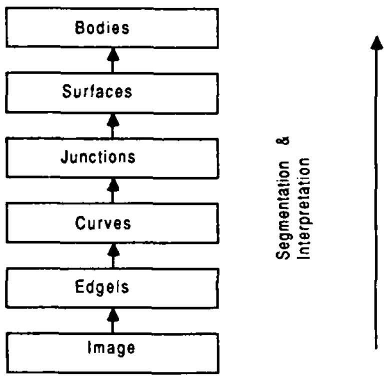


Figure 3: Hierarchical Arrangement of Structures

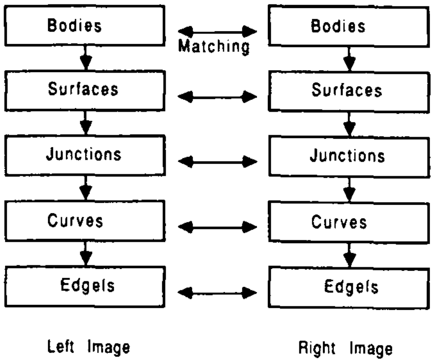


Figure 4: Hierarchical Matching of Structures

structures.

Results of implementation are shown in Section 4. These include the detection and segmentation of various structures, and how they are grouped into a hierarchical structure. Projection of matching results are given.

Conclusion and discussion are presented in Section 5.

2 Structures for Correspondence

For stereo correspondence, we desire to have structures that are invariant under general viewpoints. Viewpoint-invariant structures exhibit the same characteristics from different viewing positions and thus correspondences may easily be located. Low-level structures are usually not viewpoint-invariant. Edge angle and interval length are quasi-invariant. Contrast across an edge and gray-scale intensity are variant properties that depend on viewing positions. These variant structures are used for correspondence in previous stereo systems because there is a high correlation between the appearance of these structures in stereo images, especially when the angle between different viewpoints is small. Arbitrary thresholdings or heuristic weighting mechanisms are required to incorporate these viewpoint-dependent structures. We avoid using these viewpoint-dependent structures as primary structures *for matching*, but they certainly render useful supportive evidence when no viewpoint-independent information is available.

In this section, we will discuss the structures which we have chosen for correspondence: edgels, curves, junctions, surfaces, and bodies. Various viewpoint-independent characteristics of these structures are described. They provide useful constraints during the matching process. We also describe methods for segmenting these chosen structures from images. The segmentation process may be carried out on each image independently and they can thus be segmented simultaneously, grouped, and arranged hierarchically.

2.1 Edgels

Edgels are the low-level structures chosen. They are derived from the gray-scale intensity image and provide a good source of structures for matching. Edgels may correspond to:

- depth discontinuities (e.g. occlusion)
- continuous-surface-normal depth discontinuities (e.g. limb)
- surface-normal discontinuities (e.g. crack)
- surface-reflectance discontinuities (e.g. surface marking)

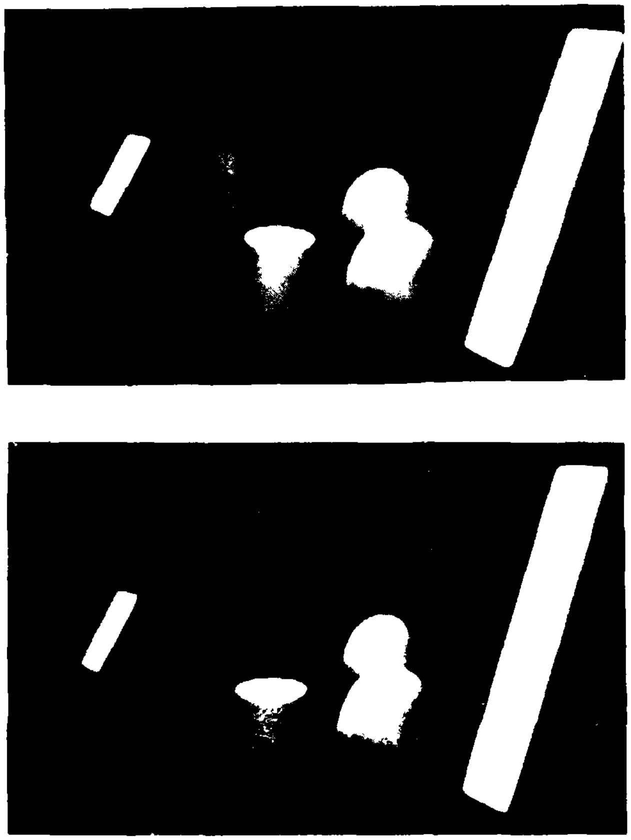


Figure 5: A Pair of Gray-scale Images (Set 1)

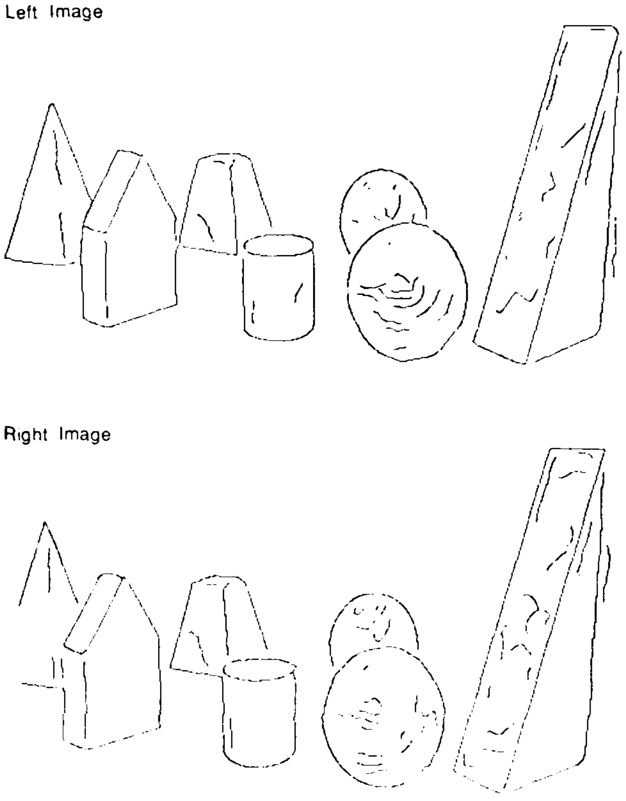


Figure 6: Edgel Detection and Curve Fitting (Set 1)

- illumination discontinuities (e.g. shadow)

The second type of discontinuities is dependent on viewpoint. Other discontinuities correspond to physical changes on surfaces in three-dimensional world. These changes are reflected in both images. Thus, they are ideal for matching.

Continuous-surface-normal depth discontinuities correspond to limbs, which are viewpoint-dependent. Thus a pair of corresponding limbs appearing in two different images do not correspond to the same physical points in space. Therefore, they should not be matched directly. A further discussion on how to utilize limbs for reconstruction of curved surfaces is presented in [Lim 1988a].

Although the detection of an edgel is an invariant, the angle of an edgel appearing in an image is not. Arnold and Binford [Arnold 1980] derive additional analytic results which calculate the likelihood that an arbitrary pair of edgels will have the same angle. They show that edge angle is quasi-invariant for stereo.

2.2 Curves

Curves are the second type of structures chosen for correspondence. Each consists of a set of ordered and connected edgels. Straight lines and planar polynomial curves preserve their orders under perspective projection, i.e. straight lines project to straight lines and planar polynomial curves project to polynomial curves in an image.

An n degree planar curve in space projects to an n degree curve in the image. In particular, conics map into conics and straight lines map into straight lines. Since the degree of a curve is preserved under projection, we can match the degree of fitted curves in a pair of stereo images. However, in practice, we need to have a very robust curve fitting algorithm and enough information in the data. A proof showing that the degree of a planar curve is invariant is given in [Lim 1987b].

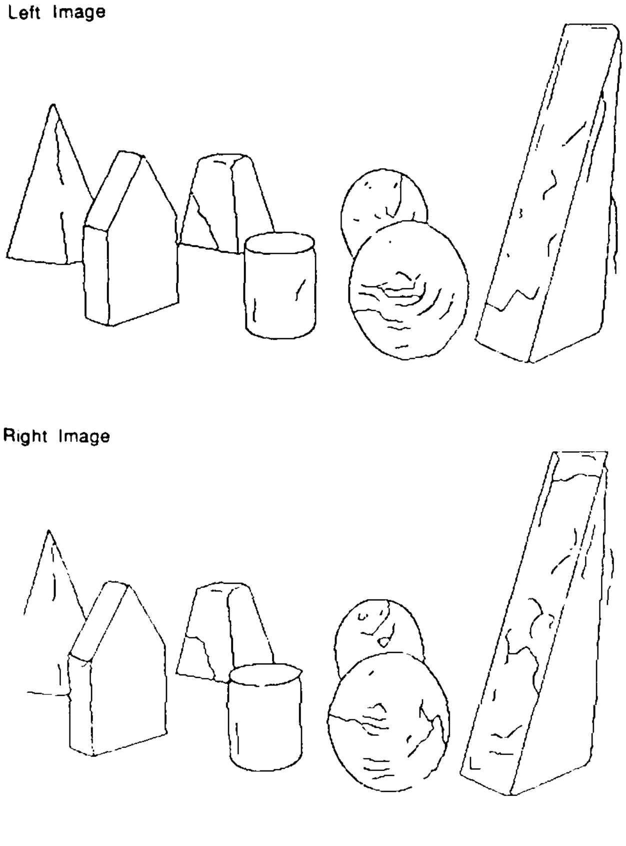


Figure 7: Curve Extension (Set 1)

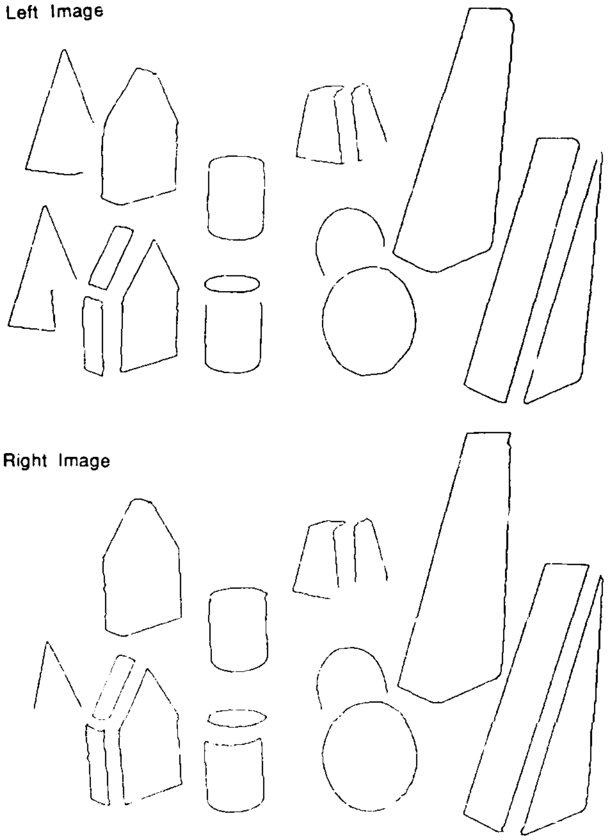


Figure 8: Segmented Surfaces (Set 1)

Binford [Binford 1981] points out that a continuous curve in space projects to a continuous curve in image. We can thus constrain a con'inuous curve in one image to match only a continuous curve in another image. Likewise, a break in a space curve (tangent discontinuity) will result in a break in its image, unless there is a coincidence in which the camera is coplanar with the two tangents of the curve at the break. Thus, breaks in image curves are good structures for matching. Because these breaks can be sharply localized, they also lead to restrictive constraints for matching.

2.3 Junctions

When two or more curves terminate at a junction, we infer that coincidence of these curves in an image implies coincidence in space. A junction in space corresponds to an intersection of two or more curves. Junctions are named according to the number of edges forming the junctions and the angles subtended by these edges (Figure 11). A junction with only two curves is called an *L* junction. A junction with three curves is called an *A* junction when one of the angles between any two curves is greater than 180 degrees, a *Y* junction when none of the angles is greater than 180 degrees, and a *T* junction when one of the angles is equal to 180 degrees. A junction with four or more edges is called an *X* junction when none of the angles is 180 degrees and a *K* junction when one of the angles is 180 degrees.

The termination of a continuous image curve (a *T* junction) can result from three different occurrences:

- occlusion of an edge (Figure 12)
- termination of a surface marking at the visible edge
- surface marking

In the first case, the formation of *T* junctions is viewpoint-dependent and the projections will not, in

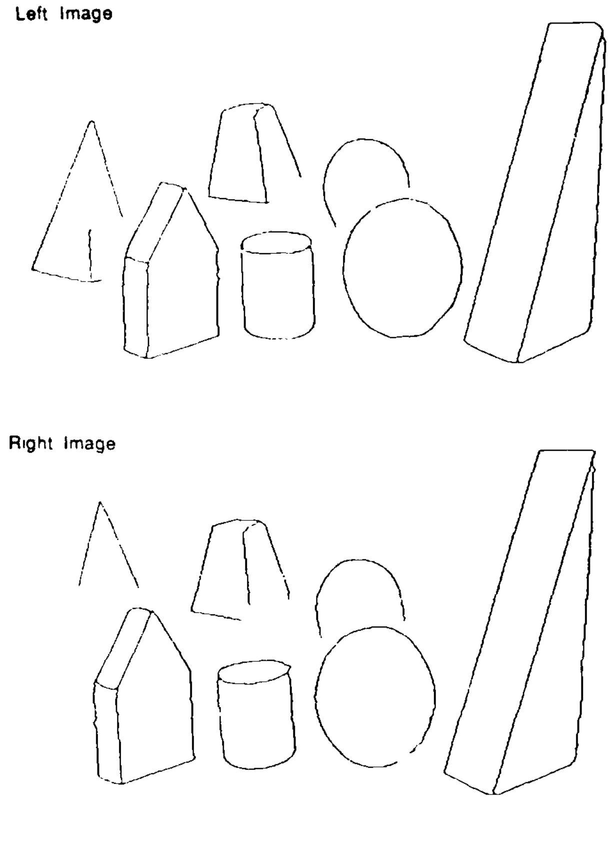


Figure 9: Segmented Bodies (Set 1)

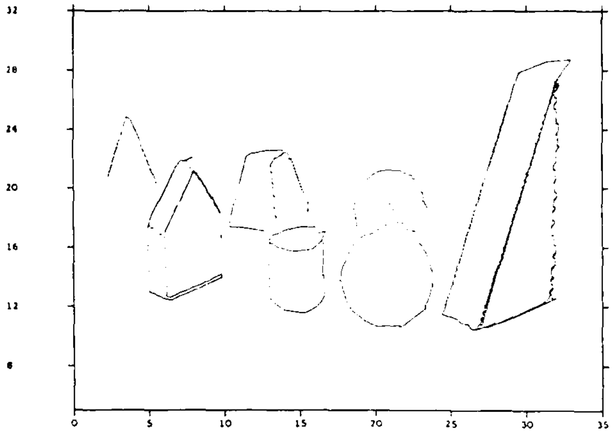


Figure 10: Projection of Reconstruction (Set 1)

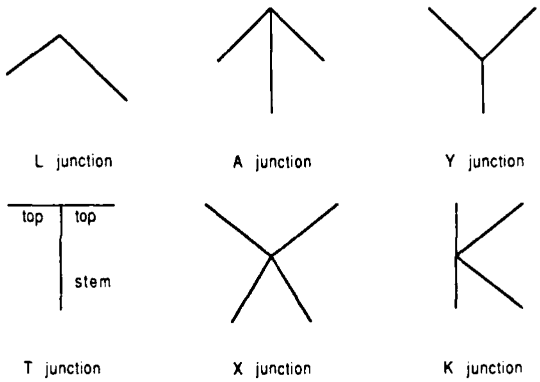


Figure 11: Classification of Junctions

general, appear on a pair of corresponding epipolar lines. We refer to the two horizontal curves in the example forming the *T* junction as the *top* and the vertical curve as the *stem* (Figure 11). The *T* junctions formed in the last two cases can usually be distinguished from the first case. In the last two cases, since the *T* junctions correspond to a physical point in three-dimensional space, the projections are viewpoint-independent and they project to a pair of corresponding epipolar lines.

A *T* junction gives strong local evidence for an occlusion [Binford 1981,Malik 1984]. Surfaces are not connected across a *T* junction. We infer that the surface bounded by the top curves occludes the other surfaces (Figure 12). For junctions of higher order, we hypothesize that any two curves forming an angle of 180 degrees belong to one surface.

2.4 Surfaces

Surfaces may be curved or planar. The image of each surface is bounded by a set of curves and junctions. Images of surfaces may have closed or open boundaries. Images of surfaces with open boundaries result when parts of the surfaces are occluded and *T* junctions are formed (Figure 12). Open boundaries may also be formed when complete information is not available, e.g. when some edges are only partially detected due to low contrast in image intensities.

2.5 Bodies

A body is a closed, connected component of surfaces in three-dimensional space. In monocular interpretation, adjacent surfaces are connected unless their shared bounding curve corresponds to a depth discontinuity. Occlusion is indicated by *T* junctions along the bound-

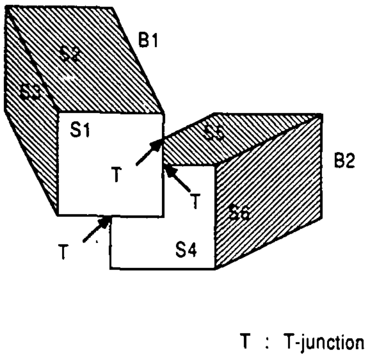


Figure 12: Occlusion of Surface and Body

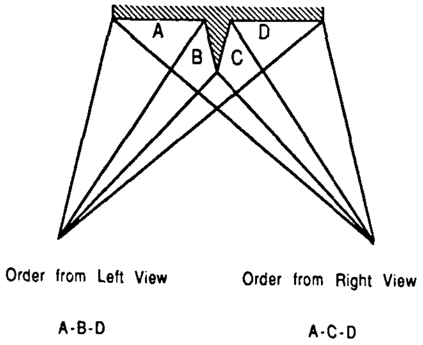


Figure 13: Order of Opaque Surfaces is Preserved

ary (Figure 12).

The order of surfaces along the baseline of the cameras is an invariant property. That is, if surface A is to the left of surface B, then this order is preserved in the images. This is true for opaque surfaces (Figure 13).

2.6 Hierarchical Arrangement of Structures

The structures detected from each image are grouped and arranged into a hierarchical structure (Figure 3). This structure is divided into five levels, from top to bottom: bodies, surfaces, junctions, curves, and edgels. The number of structures in each level is reduced as we go up the hierarchical structure (Figure 1) and the structures become more global. To assure global matching, we start matching at the highest level. The matching process is discussed in the next section.

3 Matching

In this section, we first discuss constraints which must be satisfied by matches. The matching process is then described in detail. The computational complexity of our matching scheme is compared to that of others. The result shows that the time required for matching is reduced by at least four orders of magnitude for our method.

Constraints used to limit computation and reduce ambiguity include:

- Continuity Constraint
- Disparity Constraint
- Epipolar Constraint
- Hierarchical Constraint

3.1 Continuity Constraint

Continuity of disparity along extended curves is implicit in our choice of structures. We fit extended curves to detected edgels. A pair of edgels in two images only match if the curves to which they belong are matched. Thus two pairs of matched edgels on consecutive epipolar lines must have a continuous varying disparity. This is different from the assumption of continuous disparity within epipolar lines.

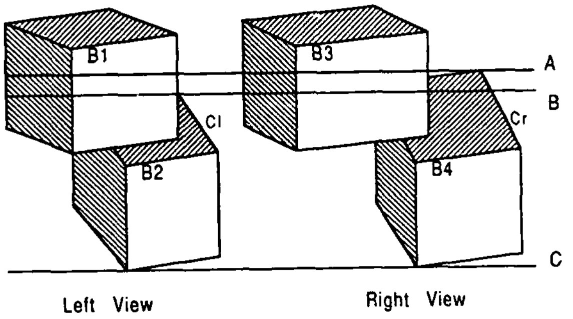
3.2 Disparity Constraint

We utilize the most general disparity constraint and assign only the weakest bounds on the disparity range. With the current set up, where the axis of the two cameras are parallel and perpendicular to the baseline, the disparity must be positive. On the average, this constraint alone reduces the search space by half, from the whole epipolar line to half of the epipolar line. This constraint has been used previously.

3.3 Epipolar Constraint

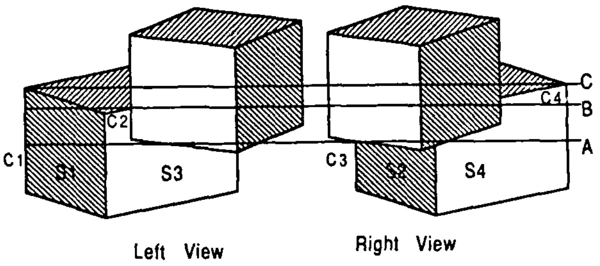
The epipolar constraint requires that points on a structure in one image must correspond to points on structures only on the corresponding epipolar line in the other image. This effectively reduces the search from a two-dimensional space to a one-dimensional space. With our camera setting, only horizontal disparity exists between a pair of corresponding structures.

We extend the epipolar constraint to curves, surfaces, and bodies. The epipolar constraint holds for every point on a curve; hence for two curves in two views to correspond, for every point on one, a corresponding



B2 matches B4; A > B

Figure 14: Inequality Constraint on Bodies



S1 matches S2; C > A
S3 matches S4; B < C

Figure 15: Inequality Constraint on Surfaces

point on the other must be found on the same epipolar plane, or the point must be obscured. Obviously, the same is true of images of surfaces which are bounded by curves, and images of bodies. This constraint involves two-dimensional information which is non-local and provides a more global construct than that which can be derived from neighborhood pixels alone.

Occlusion is identified by monocular interpretation and provides an inequality constraint on corresponding curves in the other view. They must correspond on each epipolar line or be obscured. In Figure 14, the curve Cl of body $B2$ is obscured at epipolar line A, whereas the curve Cr of body $B4$ is not. Thus we can set up an inequality constraint, requiring that

$$A > B, \tag{1}$$

$B2$ matches $B4$. In Figure 15, curve $C2$ and curve $C3$ are obscured at epipolar line B and A respectively, whereas curve $C1$ and the top portion of curve $C4$ are not. Hence we can set up two inequality constraints, requiring that

$$\begin{aligned} C &> A, \text{ and} && (2) \\ B &< C && (3) \end{aligned}$$

for the matching of $S1$ and $S2$, and $S3$ and $S4$ respectively.

3.4 Hierarchical Constraint

All the constraints introduced in the previous discussion are matching constraints. Any match must satisfy these constraints. They reduce the number of possible matches for a given structure. As a result, the total size of the search space is reduced, but it is important to note that all discussion has been independent of a particular search strategy to be employed in finding corresponding matches. The constraints hold for any search strategy.

When more than one match at a lower level is possible for a particular structure, the correspondence information at a higher level may be used to resolve this ambiguity. For example, if there is more than one match for a surface, only surfaces belonging to a pair of matched bodies are considered. This hierarchical disambiguating technique is an efficient search algorithm.

3.4.1 Matching of Bodies

A hierarchy of image structures is segmented from each image independently before the matching stage as described in the previous section. Matching (Figure 4) starts from the highest level of the structure: the body level. A pair of bodies can be matched potentially only if they have the same maximum and minimum extents. We define the *maximum* and *minimum extents* of an image structure as the maximum and minimum v-coordinates spanned by the image structure. For bodies that are partially occluded, inequality constraints are used as described above. The following simple analysis shows that the probability of two different bodies having the same extents is very small.

Assume an image with L epipolar lines and B bodies. Assume that none of the bodies are occluded. If the bodies are randomly distributed in an image then the probability of two different bodies having the same maximum extents on the same epipolar line is

$$\frac{B}{L}. \tag{4}$$

If all bodies have the same size in an image, then the probability of two different bodies having the same minimum extents on the same epipolar line is the same.

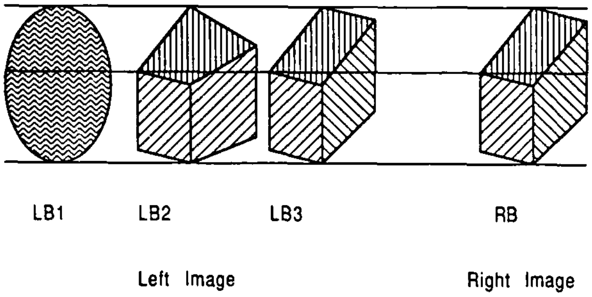


Figure 16: Multiple Potential Matches for a Body

However, different objects have different sizes and their dimensions in an image are different. Even for similar objects, their dimensions in an image are different when their orientations and distances with respect to the viewpoints are different. Assume the number of different dimensions possible for all bodies in an image is S, and that these dimensions are randomly distributed over all bodies; then the probability of two different bodies having the same dimension is given by

$$\frac{B}{S}. \qquad (5)$$

Thus, the probability of two different bodies having the same maximum and minimum extents is given by the product of Equation 4 and Equation 5, which is equal to

$$\frac{B^2}{L*S}. \qquad (6)$$

A pair of potentially matched bodies are considered matched if at least one pair of their surfaces match. If more than one body candidate matches another body in the other image, then the one with more matched surfaces is chosen. We use a simple one-level backtracking mechanism which allows backtracking if a pair of bodies is incorrectly matched. If none of the surfaces can be matched, then the potentially matched pair is considered unmatched.

In Figure 16, $LB1$, $LB2$, and $LB3$ in the left image are considered potential matches for RB in the right image because they have the same maximum and minimum extents. Since no part of surface $LB1$ matches any part of surface RB, backtracking is invoked and $LB1$ is no longer considered a potential match. $LB2$ and $LB3$ are considered matched since both of them have at least one surface which matches one surface of RB (in the sense that they both have the same extents). However, $LB3$ is finally chosen because it has more matched surfaces with RB.

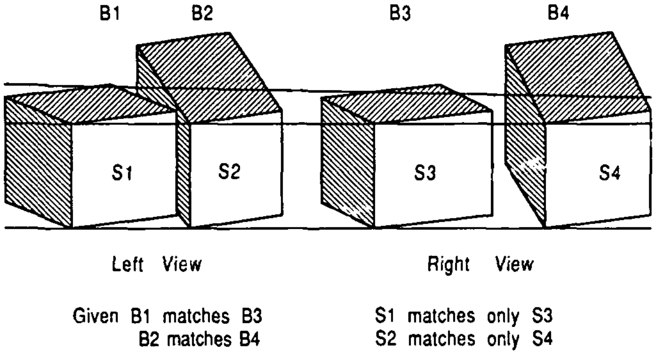


Figure 17: Matching of Surfaces

3.4.2 Matching of Surfaces

Matching a pair of surfaces from two images will only be considered if they belong to a pair of potentially matched bodies. In Figure 17, assume $B1$ matches $B3$. Without the hierarchical constraint, $S1$ can match either $S3$ or $S4$ since they span the same top and bottom epipolar lines. However, with the hierarchical constraint, only the matching of $S1$ and $S3$ is considered since they belong to a pair of matched bodies. The matching of $S1$ and $S4$ will not be considered at all. Thus the combinatorial matches between surfaces are reduced.

A pair of surfaces is considered potentially matched if they have the same extents. An analysis similar to that for bodies indicates that the probability of two different surfaces having the same maximum and minimum extents is also very small.

A pair of potentially matched surfaces is considered matched if at least one pair of their curves match (i.e. two pairs of connected junctions match). If more than one surface candidate matches another surface in the other image, then the one with more matched curves is chosen. Again we use a simple one-level backtracking mechanism which allows backtracking if a pair of surfaces is incorrectly matched. If none of the curves can be matched, then the potentially matched pair is unmatched.

In Figure 18, $LS1$, $LS2$, and $LS3$ in the left image are considered potential matches for RS in the right image because they have the same maximum and minimum extents. Since no curve of $LS1$ matches any curve of RS, backtracking is invoked and $LS1$ is no longer considered a potential match. $LS2$ and $LS3$ are considered matched since both of them have at least one curve which matches one curve of RS (in the sense that they both have the same extents). However, $LS3$

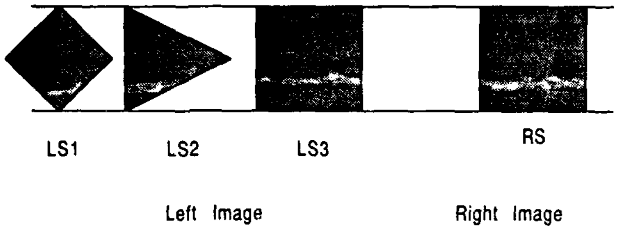


Figure 18: Multiple Potential Matches for a Surface

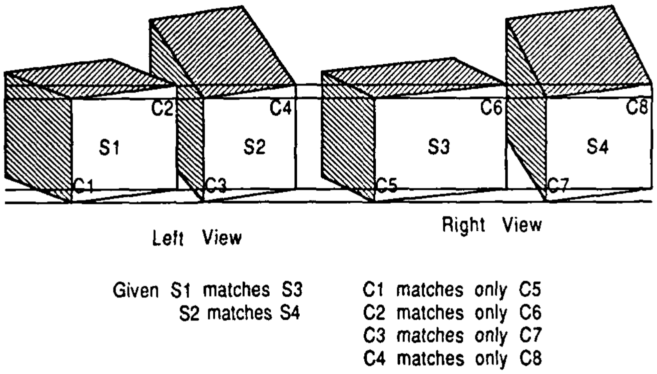


Figure 19: Matching of Curves and Junctions

is finally chosen because it has more matched curves with RS.

3.4.3 Matching of Junctions and Curves

There is a parallel between matching of surfaces belonging to a pair of matched bodies, and matching junctions and curves belonging to a pair of matched surfaces. In Figure 19 a pair of junctions and curves are matched only if they belong to a pair of potentially matched surfaces. Thus, using the hierarchical constraint and assuming $S1$ matches $S3$; only $C5$ will be considered a possible match for $C1$, and not $C7$, even though both of them have the same dimensions as $C1$. This again saves computational time by reducing possible matches. A pair of junctions are matched if they are on the same epipolar line and a pair of curves are matched if they have compatible extents.

Once a pair of surfaces is matched, the junctions and curves belonging to the matched surfaces may be matched accordingly since they are arranged in ordered lists. After matching of the curves, again the edgels belonging to the matched curves may be matched easily since they are also arranged in ordered lists.

3.4.4 Hierarchical versus Coarse-to-fine Matching

There are similarities and differences between the hierarchical approach, and the coarse-to-fine control strategy for matching discrete "interest points" used by Moravec or for matching zero-crossings proposed by Marr and Poggio.

The basic idea is to limit the search space of possible matches. In the coarse-to-fine approach, initial matching of structures uses a coarse representation of images and the density of structures is greatly reduced. The reduction of structure density reduces the search space and makes matching easier. However, this comes at the expense of possible mistakes in the initial matches. The larger the scale, the more likely it is that the support of the zero crossing will cross boundaries and that no real correspondence exists. In the hierarchical approach, we reduce the density of structures by abstraction of higher-level symbolic structures. There is no change in scale.

In both approaches, the initial matches are used to constrain the matching of finer detailed or lower-level representations. This again reduces the search space of the matching process. The estimated disparities of matching at a coarser level are used to constrain the disparity of possible matches at a finer level in the coarse-to-fine approach. The assumption of similar disparity is valid only for surfaces varying in a smooth manner relative to the viewer. In the hierarchical approach, reasoning and inference are used to interpret the structures obtained. Thus great changes in disparities are allowed and accounted for.

3.5 Complexity of Matching

We analyze the complexity of matching body extents and compare it to previous methods of area correlation and matching edgels.

3.5.1 Matching Edgels

The number of possible matches can be reduced considerably by matching only simple structures such as edgels. As we pointed out earlier, matching edgels reduces the space of possible correspondence by attempting to restrict the computation to interest points in an image.

Let there be E edgels in an image, the number of possible correspondences under a straight forward search is

$$E^2. \qquad (7)$$

The search space may be reduced by applying the epipolar constraint. Let there be L epipolar lines in each image. On the average there are E/L edgels on each epipolar line and each edgel can find E/L possible matches in the other images. Therefore, the number of possible matches within an epipolar plane is

$$\frac{E^2}{L^2}. \tag{8}$$

For the whole image, there are L epipolar lines. The total number of possible matches is

$$\frac{E^2}{L}. \tag{9}$$

We compare this result to the total number of possible matches using bodies.

3.5.2 Matching Bodies

The number of possible matches can be further reduced by matching higher-level structures. Let there be E edgels, C curves, S surfaces, and B bodies in an image after segmentation, where

$$C = \frac{E}{K_c}, \tag{10}$$

$$S = \frac{C}{K_s}, \text{ and} \tag{11}$$

$$B = \frac{S}{K_b}. \tag{12}$$

K_c, K_s, and K_b are the average number of edgels, curves, and surfaces forming a curve, a surface, and a body respectively.

Each body has a unique maximum extent. Thus the average number of maximum extents per epipolar line is

$$\frac{B}{L}. \tag{13}$$

Each of them has B/L possible matches in the other image. Therefore, the possible matches of maximum extents between a pair of epipolar lines is

$$\frac{B^2}{L^2}. \tag{14}$$

For the whole image, there are L epipolar lines and the total number of possible matches is

$$\frac{B^2}{L}. \tag{15}$$

Substitute Equations 10,...,12 into Equation 15, to get

$$\frac{E^2}{L} * \frac{1}{K_c^2 K_s^2 K_b^2}. \tag{16}$$

Comparing this to Equation 9, the matching complexity is reduced by a factor of

$$\frac{1}{K_c^2 K_s^2 K_b^2}. \tag{17}$$

For a typical image, the average number of edgels per curve is around 30, the average number of curves per surface is around four, and the average number of surfaces per body is around three. Thus, we have a reduction of more than four orders of magnitude in computation time. The reduction in computation time is gained at the expense of segmentation and monocular interpretation. However, the segmentation and interpretation also provide additional meaningful results on the recovered depth data.

4 Implementation and Results

4.1 Edgel Detection and Curve Fitting

A stereo pair of gray-scale images is obtained from a CCD camera (Figure 5). Edgels are detected by applying Nalwa's directional edge operator [Nalwa 1986] to the pair of images. These edgels are first aggregated into ordered sets corresponding to individual extended edges. Straight lines and conic sections are then fitted to these edgels in a segment in a best-fit sense [Nalwa 1987]. Position and tangent continuity are preserved at this stage (Figure 6).

4.2 Curve Extension

Nalwa's edge operator is designed to detect a single step-edge profile within each window. When more than one edge appears within the same window, the edge operator misses edges and estimates edge parameters inaccurately. In particular, edges are not properly detected around junctions. Thus, the curves obtained by the curve-fitting process are usually incomplete near junctions where three or more curves meet. Only a few junctions are found at this stage. However, the information of junctions is critical to the segmentation of surfaces and bodies.

We extend curves near junctions in order to estimate junctions. The tangent of a fitted curve near a junction is a good indication of the orientation of the nearby missing edgels. The contrast across the fitted curve also gives a good estimate of the contrast across the missing edgels.

A directional difference operator is chosen to detect the missing edges around junctions. This operator is applied to regions where end points of curves are not connected to any junction. The direction and contrast across the newly detected edgels are constrained by the direction of the tangent of the fitted curve and the contrast across the curve. The direction of the new edgels must be within $\pi/4$ radians of the tangent of the fitted curve and the contrast across the new edgel must be at least half of the original contrast across the fitted curve. Therefore the contrast required for the detection of a missing edgel is dynamically adjusted according to previously detected edgels, it is not set a priori. This smaller operator operates on the intensity values of the underlying gray-scale image.

Curves with length less than a fixed threshold are not extended. This avoids extensions on noisy curves in the image. Figure 7 show the results after curve extension.

4.3 Junction Classification

New junctions formed by curve extension and old junctions obtained directly from curve fitting are classified by their types and orders as described in Section 2.

In the implementation, because of noise in the data, we allow a variation of five degrees for determining T junctions and K junctions where one of the angles should be 180 degrees.

4.4 Surface Segmentation

The images of surfaces have two kinds of boundaries: open surface boundaries and closed surface boundaries. An open surface boundaries may occur if a surface is occluded or if a surface has missing edges.

Tracing surface boundaries is carried out by left-wall (or right-wall) following. For left-wall following, the tracing starts at a junction. It follows a curve of that junction and takes the closest counter-clockwise curve whenever it arrives at another junction (Figure 20). Exceptions arise when the tracing comes to a T junction or a K junction. If the last curve that is being traced is tangent to any other curve, then it will take the tangent curve instead of the closest counter-clockwise curve. Also, if the closest counter-clockwise curve is tangent to another curve, then the closest-clockwise curve will not be chosen. This is based on the assumption that curves that are tangent to each other belong to the same surface (Figure 21).

Tracing which starts from the stem of a T junction must end at the stem of another T junction or a junction of order one, forming an open surface, e.g. surfaces $S2$ and $S4$ in Figure 20. Tracing which starts from a junction of order one can stop at another junction of order one or at the stem of a T junction. If the tracing starts at an A or Y junction, it stops when it comes back to the starting junction, making a loop, e.g. surfaces $S1$ and $S3$ in Figure 20. Thus a closed surface is formed. If it comes to a T junction from the stem or a junction of order one, then no surface is segmented. Any surface boundary which ends at a stem of a T junction or a junction of order one is taken care of by tracings which start from the stem of a T junction. For example, the tracing which starts at junction A and follows curve C comes to a T junction but no surface is formed. Instead surface $S4$ is formed when the tracing starts from the stem of a T junction.

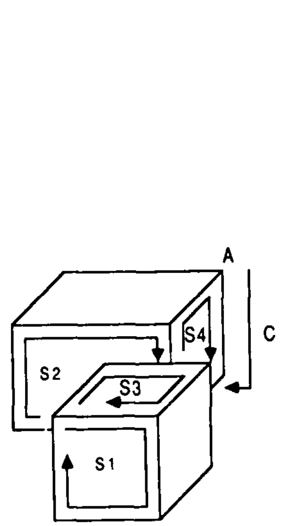


Figure 20: Tracing Surfaces

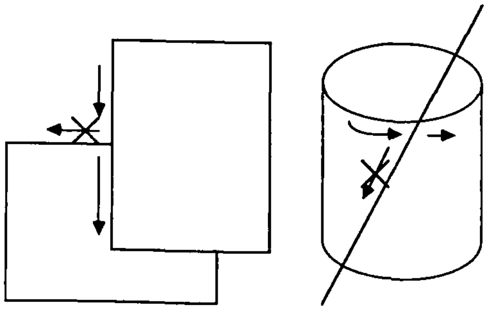


Figure 21: Tracing Surfaces with T and K Junctions

The tracing process repeats itself for every curve of a junction and for every junction. After tracing curves belonging to all junctions, there are still a few curves untraced. They are those belonging to closed surfaces that do not have any junctions, e.g. a projection of a sphere. All the curves that have been traced are marked and any curves that remain untraced are then traced. Surfaces that do not enclose more than 180 degrees are not considered. This eliminates noisy data such as some isolated curves in the center of the larger sphere in Figure 7. The results of surface segmentation on Figure 7 are shown in Figure 8. Each surface is offset for clarity.

4.5 Body Segmentation

A body is a connected component of surfaces in three-dimensional space. Surfaces which are coincident are grouped as bodies, i.e. for which there is evidence for coincidence. A surface which occludes another surface is not coincident with it across the occluding boundary, although they may be coincident along another boundary. Imperfect surface segmentation can lead to grouping of more than one object into a body. Bodies form the highest level of segmentation in the hierarchical structure.

Results from grouping surfaces are shown in Figure 9. The small sphere at the back and the truncated pyramid are not joined with objects in front because of inference of occlusion from T junctions. Again the bodies are offset for clarity.

4.6 Matching Results

Bodies, surfaces, curves, junctions, and edgels extracted from the stereo pair of images are then used for matching as described in Section 3. The projection of three-dimensional depth data in shown in Figure 10.

It is important to note that the range data recovered is already segmented into three-dimensional surface boundaries and body boundaries. For each point of depth data recovered, we know that it belongs to a specific curve of a specific surface of a specific body in space. There are symbolic representations attached to the recovered depth data. Using these segmented data, it is easy to perform surface interpolation.

In previous approaches, where edges are matched, the recovered three-dimensional raw data are isolated data with no meaning attached to them. Additional segmentation efforts in three-dimensional space are necessary for extracting symbolic information from the raw range data. Fitting surfaces to neighboring edges may not be meaningful, since the edges may not belong to the same surface.

4.6.1 Additional Results

Results from another set of images and their results are shown in Figure 22,...,27.

4.6.2 High Resolution Results

Under the current implementation, the vertical position (with respect to the image plane) of a detected edgel is quantized to the nearest epipolar line. This results in a simpler and faster matching process. However, accuracy is lost and some of the recovered depth data have jagged boundaries. This can be seen in the octagonal block on the lower right of Figure 27. We show a close-up view of the octagonal surface in Figure 28.

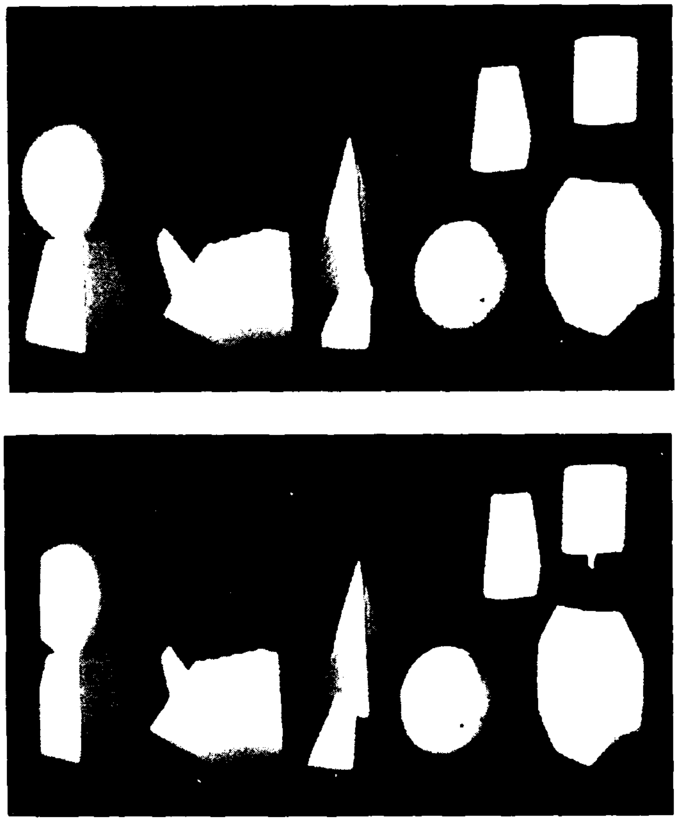


Figure 22: Another Pair of Gray-scale Images (Set 2)

To recover more accurate results, one can do an interpolation on the detected edgels or increase the resolution of edgel position by increasing the number of epipolar lines. We choose to study the latter method.

We increase the resolution of the detected edgels around the octagonal block by five times, i.e. the number of epipolar lines spanned by the octagonal surface is increased by five-fold. The matching result is shown in Figure 29. It can be seen that the boundary of the octagonal surface is as smooth as the boundary in the original gray-scale image. This indicates that the depth data recovered is very accurate.

Another example is shown using the sphere in Figure 27. The close-up view of the sphere is shown in

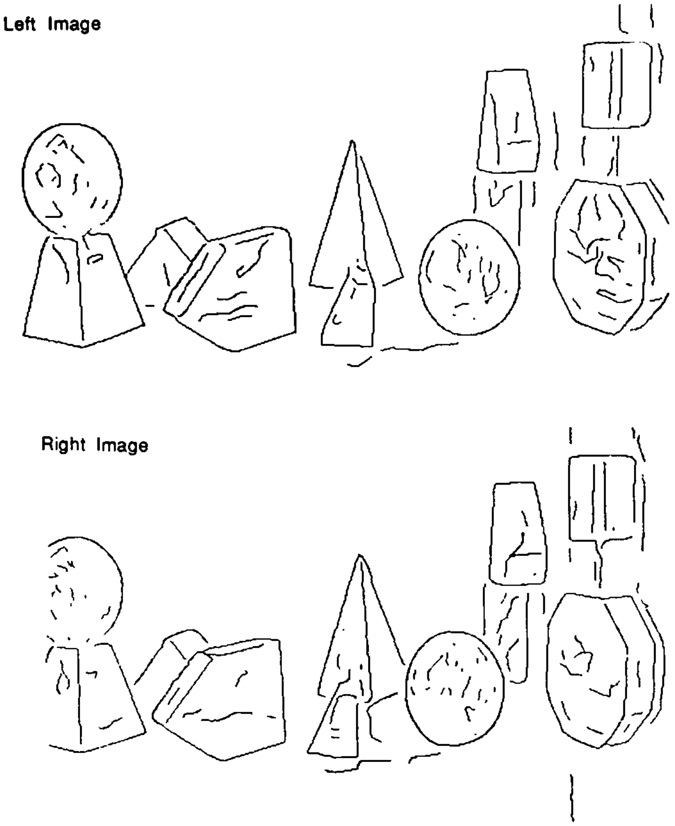


Figure 23: Edgel Detection and Curve Fitting (Set 2)

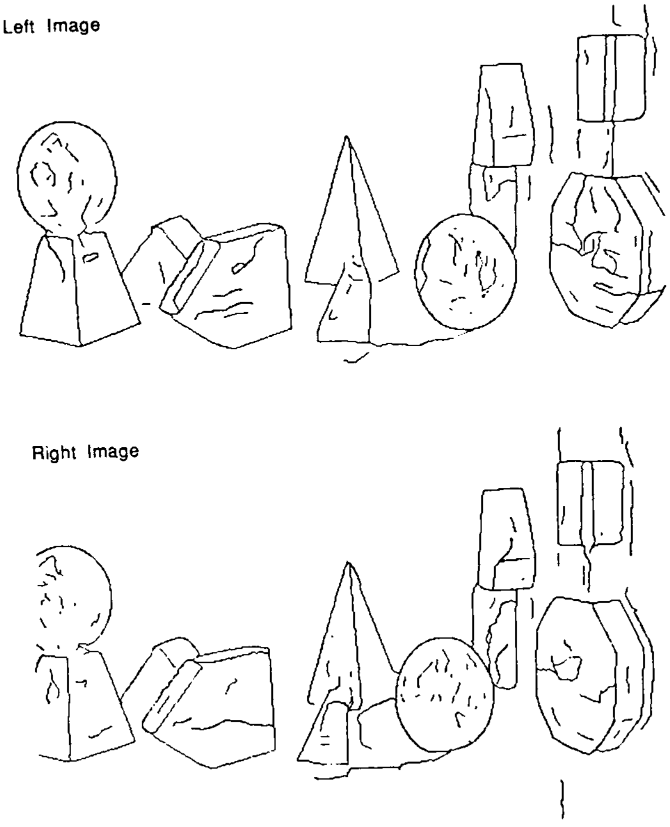


Figure 24: Curve Extension (Set 2)

Figure 30.

Again, we increase the resolution of the detected edgels locally around the sphere by five times. The matching result is in Figure 31.

These results indicate that the jagged boundaries in the projection of the results are due to the quantization errors introduced in the implementation and not the segmentation or matching processes.

5 Conclusion and Discussion

The system performs a high-level stereo correspondence which achieves global correspondence by means of high-level structures: surfaces and bodies. The main domains of application are in images where there are monocular structures. The performance of the system is based on its ability to extract these structures for matching. It utilizes geometric constraints, interpretation on occlusion and coincidence, and knowledge of surface and body.

The system is not designed for random-dot stereograms where no monocular structure can be found in the images. Its performance is also limited on natural scenery where the structures of objects are not well-defined. We have tried our system on a pair of aerial photos of urban area. We experience problems during the segmentation stage. Regions in natural scenery usually do not have well-defined structures and that makes segmentation difficult. A leaf may have a well-defined boundary but not a tree. The system is not able to segment meaningful structures from the detected edgels for matching. Textured surfaces also create problems for segmentation. The edge operator tends to detect edgels on a texture surface, but the curve fitting process cannot distinguish between edgels forming the boundaries of a surface and others that arise from surface texture.

We showed that by segmentation and monocular interpretation, we are able to match surfaces and bodies.

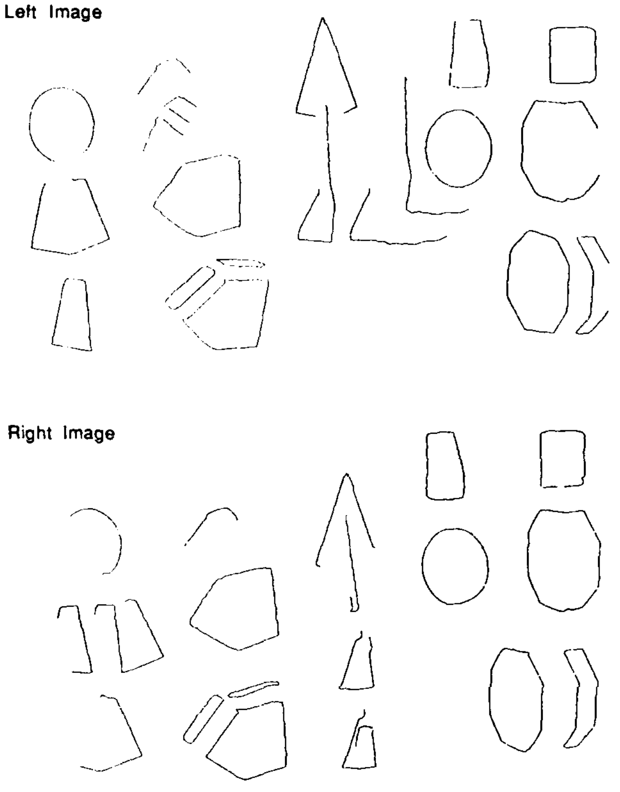


Figure 25: Segmented Surfaces (Set 2)

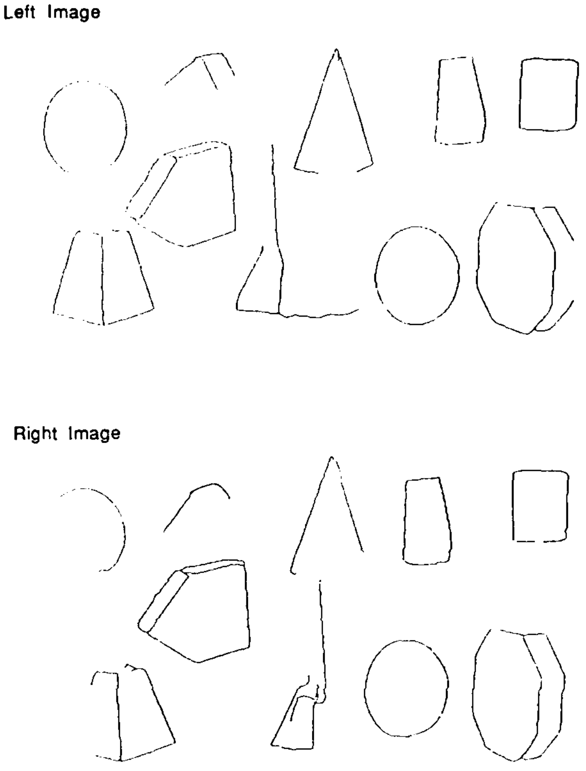


Figure 26: Segmented Bodies (Set 2)

This results in reduction of computational complexity and avoidance of local mismatches. Since the structures for matching are segmented, the recovered three-dimensional depth data are also segmented. This is very useful for surface interpretation. In contrast, previous approaches use sparse edgels for correspondence and segmentation effort concentrates on the recovered sparse depth data.

References

[Arnold 1980] R. D. Arnold and T. O. Binford. Geometric constraints in stereo vision. In *Proceedings: Society of Photo-Optical Instrumentation Engineers, Vol. 238*, pages 281-292, 1980.

[Arnold 1981] R. D. Arnold. *Automated Stereo Perception*. PhD thesis, Stanford University, March, 1981.

[Binford 1981] T. O. Binford. Inferring surfaces from images. *Artificial Intelligence, Vol. 17*, 205-244, August, 1981.

[Lim 1987a] H. Lim and T. O. Binford. Stereo Correspondence: A Hierarchical Approach. In *Proceedings: Image Understanding Workshop*, 1987.

[Lim 1987b] H. Lim. Stereo Vision: Structural Correspondence and Curved Surface Reconstruction. PhD thesis, Stanford University, September, 1987.

[Lim 1988a] H. Lim and T. O. Binford. Curved Surface Reconstruction using Stereo Correspondence. In *Proceedings: Image Understanding Workshop*, 1988.

[Malik 1984] J. M. Malik and T. O. Binford. A theory of line drawing interpretation. In *Proceedings: Image Understanding Workshop*, pages 188-194, October, 1984.

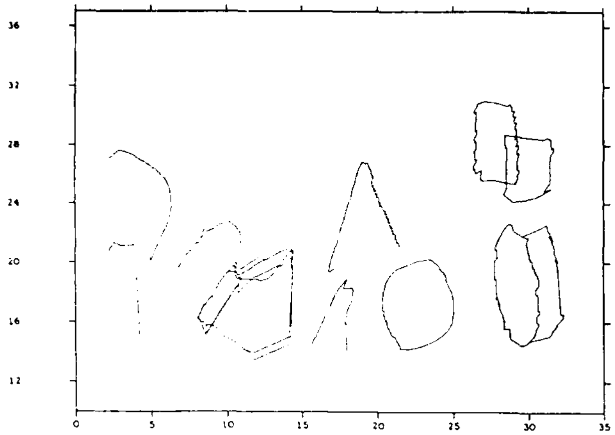


Figure 27: Projection of Reconstruction (Set 2)

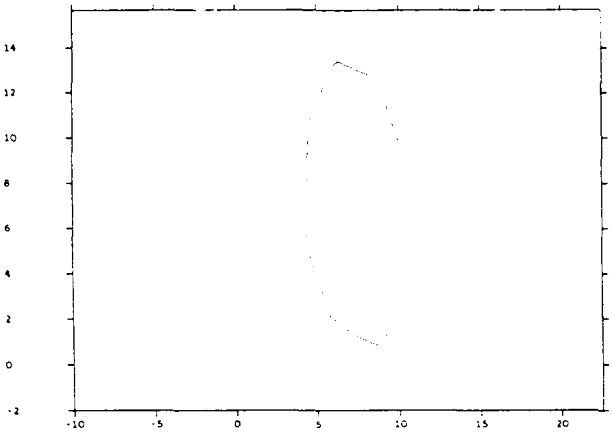


Figure 29: Recovered Octagonal Surface from High-resolution Data

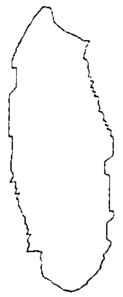


Figure 28: Close-up View of the Octagonal Surface

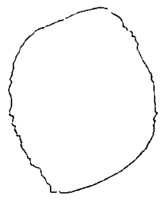


Figure 30: Close-up View of the Sphere

[Nalwa 1986] V. S. Nalwa. On detecting edges. *IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 8*, 699-714, November, 1986.

[Nalwa 1987] V. S. Nalwa and E. Pauchon. Edgel-aggregation and edge-description. *Computer Vision, Graphics, and Image Processing, Vol. 40*, 79-94, October, 1987.

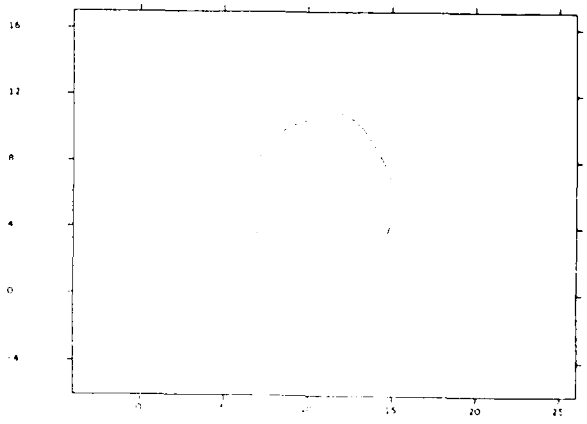


Figure 31: Recovered Sphere from high-resolution Data

CURVED SURFACE RECONSTRUCTION USING STEREO CORRESPONDENCE*

Hong Seh Lim[†] and Thomas O. Binford

Robotics Laboratory, Computer Science Department,
Stanford University, Stanford, CA 94305, U.S.A.

Abstract

An inherent problem in stereo correspondence is that apparent boundaries of a curved surface from a pair of stereo images do not correspond to the same physical points on the curved surface; therefore, they cannot be matched directly. In this paper, we describe a high-level stereo vision system which replaces the inappropriate correspondence constraint with an accurate constraint that the curved surface is tangent to lines of sight along four rays on each epipolar plane. We also derive equations from apparent boundaries of a curved surface in both images in order to reconstruct an approximation to the curved surface in space. The terminator, an edge bounding the curved surface, is used as an additional constraint whenever it is available. This is particularly applicable to objects that can be described in terms of generalized cylinders. Surface maps are obtained from reconstructed curved surfaces; thus, surface interpolation is not necessary.

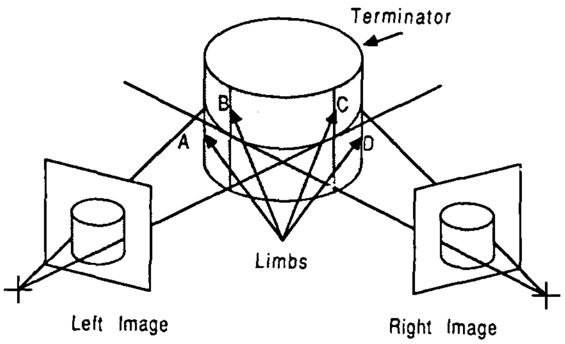


Figure 1: Stereo Views of a Curved Surface

1 Introduction

Apparent boundaries of curved surfaces appear where surface normals are perpendicular to the line of sight, i.e. where the surface is tangent to the line of sight. We refer to these apparent boundaries as *limbs* (Figure 1). Limbs are distinct from "true" edges of surfaces which correspond to a discontinuity of surface normal at the intersection of surfaces. For a limb, the curve in space which projects to the image curve depends on viewpoint. On the other hand, true edges are viewpoint-independent. At a limb, the surface has continuous tangent. Limbs are distinguished from edges which are discontinuities in surface reflectivity. Limbs coincide with pigment boundaries, which are viewpoint-independent, only from a constrained viewpoint, i.e., only on a set *of measure zero* [Binford 1987].

In Figure 1, the boundaries of a cylinder as seen in the left image correspond to curves A and C on the surface of the cylinder, while the boundaries of the cylinder as seen in the right image correspond to curves B and D. Thus, curves from a pair of images of corresponding limbs do not correspond to the same physical curve in the world; therefore we cannot recover the three-dimensional depth data by simple triangulation. It is frequently proposed to take the intersection of the tangents as the "edge". But there are only two of those which allow only a planar approximation. By using accurate constraints we obtain tangent and curvature information which cannot be obtained by a planar approximation.

A *terminator* of a curved surface is an edge bounding the curved surface. In Figure 1 the terminator is an ellipse.

*This work was supported by the Defense Advanced Research Projects Agency under contract number N00039-84-C-0211

[†] Present Address: IBM Corporation, Los Angeles Scientific Center, 11601 Wilshire Boulevard, Los Angeles, CA 90025

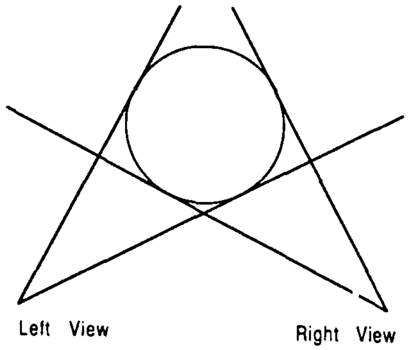


Figure 2: Planar Cross-section of a Curved Surface

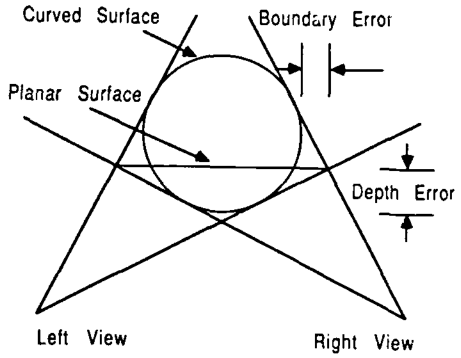


Figure 3: Errors of Fitting Planar Surface

We design and implement a solution for reconstructing objects with curved surfaces. Curved surface reconstruction has been treated very little [Shapira 1978]. The problem is that corresponding limbs extracted from stereo images are not the same points on a curved surface as described above, and they cannot be matched directly. In the next section, we show that the errors which are introduced by fitting a planar surface to incorrect depth data instead of fitting a curved surface using physical constraints derived from the camera geometry. In Section 3, we show that the lines of sight from stereo views can restrict the recovered cross-sections of a curved surface to one degree of freedom. Additional constraints are employed to determine the cross-sections uniquely. Each curved surface is then reconstructed by a number of planar conics. Results of implementation are shown in Section 4.

2 Errors between Fitting Planar and Curved Surfaces

Consider an epipolar plane cutting a curved surface to form a planar cross-section. This cross-section is tangent to the four lines of sight, two from the left view and two from the right view (Figure 2).

There are two types of error in fitting a planar surface instead of a curved surface to image data (Figure 3): the error between the true and apparent boundaries of the reconstructed surfaces, and the error between the true and apparent depths of the reconstructed surfaces.

2.1 Boundary Error

Consider part of a cross-section of a curved surface in Figure 4. The lines of sight from left and right views are tangent to the cross-section. If we consider the intersection of two corresponding lines of sight to be from the same physical point, the apparent boundary of the cross-section is at A. However, the true boundary of the cross-section is at B instead.

The error between the true boundary and the apparent boundary is shown in [Arnold 1981] to be equal to

$$r * (sec(\frac{a}{2}) - 1), \qquad (1)$$

where r is the radius of curvature of the cross-section and a is the angle between the two lines of sight. It is interesting to note that the wider the angle between the two viewpoints, the larger the error. This is in contrast to random error in triangulation which is smaller when the angle between the viewpoints becomes wider. In this case, the error is systematic.

2.2 Depth Error

Consider the error in depth between a curved surface and its planar approximation. For simplicity of calculation, we assume that the cross-section of the curved surface is a circle and that its center is located at a distance d from both cameras (Figure 5). The distance between the apparent boundary, A, and the true boundary, B, can be shown to be

$$r(1 - r\frac{sec(\frac{a}{2})}{d}), \qquad (2)$$

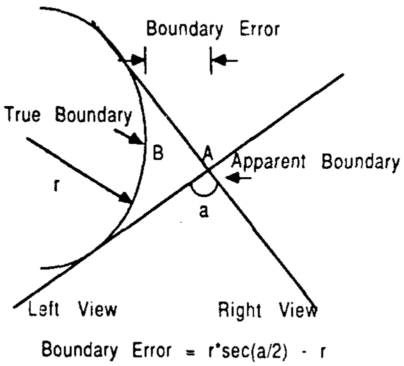


Figure 4: Error of Apparent Boundary

where r is the radius, a is the angle between the two lines of sight, and d is the distance between either viewpoint and the center of the circle.

From the equation, we observe that the farther away the object, the greater the error. The maximum error equals the radius of curvature of the curved surface when the object is located at infinity. Contrary to the boundary error, the depth error decreases slightly as the angle between the two viewpoints increases.

Proof:

Referring to Figure 5, circle O with radius r is at an equal distance d from left camera L and right camera R. The angles subtended by the circle at both cameras are e. Line $\overline{RS}$, $\overline{LP}$, and $\overline{LQ}$ are tangent to the circle O at S, P, and Q respectively. The angle between the two corresponding lines of sight $\overline{RT}$ and $\overline{LT}$ is a. Line $\overline{OT}$, of length m, bisects angle $\angle POS$ while line $\overline{OB}$, of length l, bisects angle $\angle QOS$. Thus,

$$\begin{aligned} \angle POT &= \angle TOS & (3)\\ &= c, \text{ and} & (4)\\ \angle SOB &= \angle BOQ & (5)\\ &= b. & (6) \end{aligned}$$

Angle $\angle POS$ equals angle $\angle LTR$, therefore,

$$\angle POT = \angle TOS = c = \frac{a}{2}. \qquad (7)$$

In triangle OPT,

$$\begin{aligned} m &= r * sec(c) & (8)\\ &= r * sec(\frac{a}{2}). & (9) \end{aligned}$$

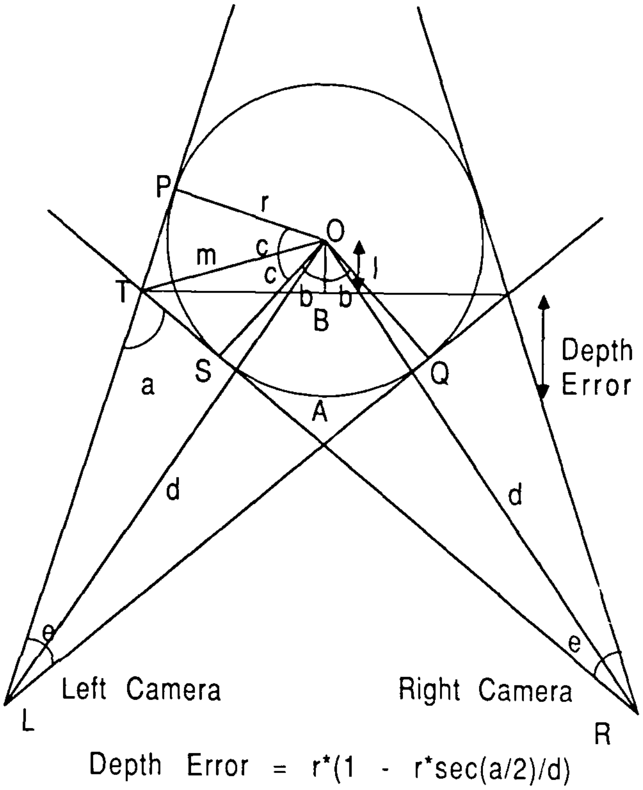


Figure 5: Error of Apparent Depth

Consider quadrilateral $OQLP$, angle $\angle OQL$, angle $\angle QLP$, angle $\angle LPO$, and angle $\angle POQ$ add up to 360 degrees, i.e.,

$$90 + e + 90 + (2c + 2b) = 360, \qquad (10)$$

In triangle OBT,

$$l = m * cos(c + b), \qquad (11)$$

and in triangle OLP

$$d * sin(\frac{e}{2}) = r. \qquad (12)$$

Substituting Equation 9 and 10 into Equation 11, we have

$$l = r * sec(\frac{a}{2}) * sin(\frac{e}{2}). \qquad (13)$$

Eliminating $sin(\frac{e}{2})$ between Equation 12 and 13, we get

$$l = \frac{r^2 sec(\frac{a}{2})}{d}, \qquad (14)$$

and thus the depth error is equal to

$$r - l = r(1 - r\frac{sec(\frac{a}{2})}{d}). \qquad (15)$$

3 Curved Surface Reconstruction

3.1 Fitting Conics to Four Lines of Sight

To reconstruct a curved surface from a pair of stereo images, we first cut the curved surface into a number of slices, each slice in an epipolar plane. Within each epipolar plane, there are four lines of sight, two from each viewpoint. Each line of sight is tangent to the cross-section of the curved surface. We fit a conic tangent to these four lines of sight in each epipolar plane. Combining all the reconstructed conics in epipolar planes gives the curved surface.

We show that the equation of a conic tangent to four given lines is given by

$$L^2E^2 - 2L(AC + BD) + F^2 = 0, \qquad (16)$$

(this abridged notation is explained in the proof) where A, B, C, and D are the four tangent lines, E and F are the diagonals of the quadrilateral formed by the four tangent lines,

$$AC - BD = EF, \qquad (17)$$

and L is a parameter that can be chosen freely.

Proof:

We adopt the abridged notation in [Salmon 1904], where an equation of a conic

$$ax^2 + bxy + cy^2 + dx + ey + f = 0 \qquad (18)$$

is represented by

$$S = 0, \qquad (19)$$

i.e. S stands for $ax^2+bxy+cy^2+dx+ey+f$. Similarly, an equation of a straight line

$$ax + by + c = 0 \qquad (20)$$

is represented by

$$A = 0, \qquad (21)$$

i.e. A stands for $ax + by + c$. Let $S = 0, T = 0$, and $U = 0$ be conics, $A = 0, B = 0, C = 0, D = 0, E = 0$, and $F = 0$ be straight lines, and K and L be constants.

We know that if $U = 0$ and $V = 0$ are the equations of any two loci, then the locus represented by the equation $U + kV = 0$ (where k is any constant) passes through every point common to the two given loci. Assume that A and B intersect S at four points m, n, o, and p, as illustrated in Figure 6. It is obvious that a conic T passing through these four intersection points is given by

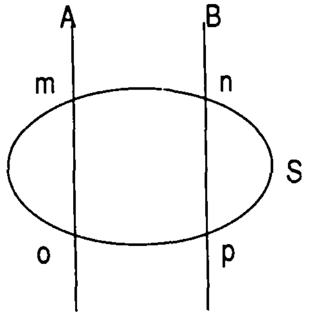


Figure 6: Lines A and B Intersect Conic S at Four Points

$$T = S - KAB, \qquad (22)$$

(Figure 7).

Suppose that lines A and B get closer and closer to each other until they coincide, then point m coincides with point n, and point o coincides with point p. The conic T now touches the first conic S at m and o (Figure 8). Thus, Equation 22 becomes

$$T = S - KA^2 \qquad (23)$$

which represents a conic tangent to S at two distinct points. A is the chord of contacts.

Next, we want to find an equation of a conic tangent to two given conics S and T. Assume that S and T intersect at four points m, n, o, and p. E and F are their chords of intersection (Figure 9), i.e.

$$T = S - EF. \qquad (24)$$

From Equation 23, a conic tangent to S can be represented by

$$4LS - (LE + F)^2 = 0, \qquad (25)$$

where $LE + F$ is the chord of contacts between the new conic and conic S, and L is any constant. Similarly, a conic tangent to T can be represented by

$$4LT - (LE - F)^2 = 0, \qquad (26)$$

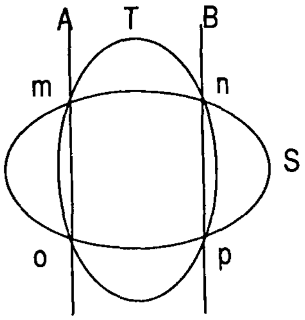


Figure 7: Conic T Passes through Four Points of Intersection

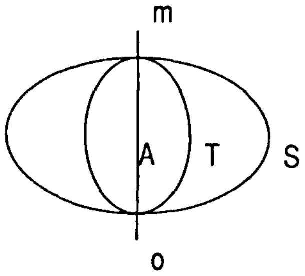


Figure 8: Conic T Tangent to Conic S

where $LE - F$ is the chord of contacts between the new conic and conic T. Combining Equation 25 and Equation 26, we have

$$L^2E^2 - 2L(S+T) + F^2 = 0, \tag{27}$$

which represents a conic tangent to both S and T (Figure 9).

When the two conics S and T degenerate into two pairs of straight lines A, C and B, D, we have

$$\begin{aligned} S &= AC \text{ and} & (28)\\ T &= BD, & (29) \end{aligned}$$

then Equation 27 and Equation 24 become

$$\begin{aligned} L^2E^2 - 2L(AC+BD) + F^2 &= 0, \text{ where} & (30)\\ AC - BD &= EF. & (31) \end{aligned}$$

3.2 Curved Surface Reconstruction with Extremum Constraint

From Equation 30, we note that there is a free parameter L which can assume any value. This is to be expected as a general conic is defined by five parameters and we have only four constraints, namely, the four tangent lines from the four lines of sight.

To determine the fifth parameter, without additional information, we employ the extremum principle suggested by Brady and Yuille [Brady 1983]. It maximizes a familiar measure of compactness or symmetry of surface, namely the ratio of the area to the square of the perimeter. Specifically, the measure is

$$M = \frac{Area}{Perimeter^2}. \tag{32}$$

This measure defines characteristics of a curve which are independent of the scale, and for all possible curves it is maximized by the most symmetric one, a circle. The measure has an upper bound of $\frac{1}{4\pi}$ when the curve is a circle. Its lower bound, zero, is achieved when the curve becomes a straight line.

3.3 Curved Surface Reconstruction with Terminator Constraint

To determine the fifth parameter, we may use other information from the image, such as specularity, markings, and texture on a curved surface, or the terminator of a curved object. Specularity provides viewpoint-dependent information while surface markings, surface texture, and the terminator render viewpoint-independent information. Recall that the terminator is the intersection of the surface with another, as in Figure 1.

For linear homogeneous generalized cylinders, we can use the terminator to constrain the fifth parameter. In particular, if the terminator is a conic, the fifth constraint may be the ratio of the major axis to the minor axis of the terminator. The ratio is kept constant for each of the reconstructed conics while the size of the cross-section may vary.

Surface markings and surface texture may provide additional constraints to each of the reconstructed con-

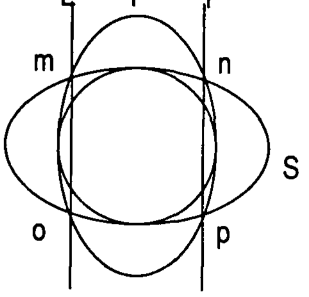


Figure 9: A Conic Tangent to Two Conics S and T

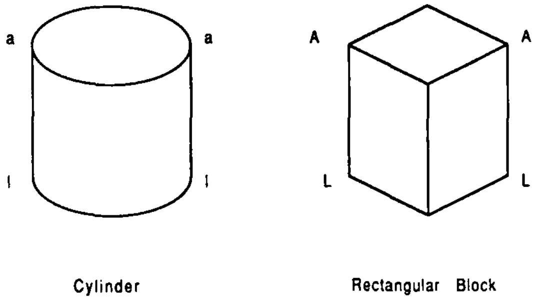


Figure 10: Distinctions between Curved and Planar Surfaces

ics. Within each cross-section of the curved surface, if a pair of corresponding surface markings can be identified, the three-dimensional depth of that marking may be determined. Each provides an additional constraint for uniquely determining the conics.

4 Results of Implementation

In this section, we provide examples of reconstruction of curved surfaces. As discussed in Section 3, we find a best-fit conic in every epipolar plane that cuts a curved surface. By combining these fitted conics, we reconstruct the curved surface. When limbs are the only information available, we use the extremum constraint which maximizes the ratio of the area to the square of perimeter of a conic. When additional information such as the terminator is available, we may use the ratio of the major axis to the minor axis of the terminator as an additional constraint.

In the current implementation, curved surfaces are not segmented automatically but by hand although segmentation could have been automated. We can distinguish a curved surface from a planar surface provided we can identify the junctions correctly. In Figure 10, if we can distinguish between junction a and junction A, or junction l and junction L, then we can identify the the curved surface. In practice, it may be difficult to make the distinction because of noisy data.

4.1 Extremum Constraint

Figure 12 provides a close-up look at a pair of corresponding cylinders appearing at the top right corner of the left and right gray-scale images in Figure 11. The corresponding segmented results are shown in Figure 13. The segmentation process from a gray-scale image is described in [Lim 1987].

Within each epipolar plane, we fit a conic to the four lines of sight and apply the extremum principle to constrain the fifth parameter. When all the fitted conics are stacked together, a curved surface is recovered which is shown in Figure 14. Projection of the same curved surfaces from a higher angle is shown in Figure 15.

The limbs of the reconstructed cylinder appear irregular. The dimensions of the reconstructed conics are very sensitive to the data of the limbs in the stereo images. The quantization errors of the limbs in the images contribute to the quantum jumps of the limbs in space.

Figure 16 is a close-up view of a stereo pair of matched spheres appearing at the center left of both images in Figure 11. The segmented results appear in Figure 17.

Again the extremum principle is applied to constrain the fifth parameter. The recovered conics are constrained to be as circular as possible. The results from different viewpoints are shown in Figure 18 and 19.

4.2 Terminator Constraint

When the terminator is available, an additional constraint can be derived from it. From a pair of matched terminators, we calculate the depth of every pair of corresponding edgels belonging to the terminator. A set of three-dimensional depth data is obtained. The data are fitted with a planar surface and then a conic. The ratio of the major axis to the minor axis is calculated and used as an additional constraint.

Figure 21 shows a pair of cylinders with observable terminators which appeared in Figure 20.

Results from fitting conics to each cross-section of the curved surface and stacking them together a:^ shown in Figure 22. The curved surface in Figure 23 is viewed from a different angle.

Figure 24 shows the result of fitting conics to the same data using the extremum constraint instead of the terminator constraint. It is observed that the limbs in Figure 23 appears to be more regular and less susceptible to the quantization errors of the limbs in the images. The extremum constraint maximizes the ratio of area to the square of perimeter for each conic independently whereas the terminator constraint requires all the conics to have the same ratio of major axis tc minor axis for all the conics. Thus the terminator provide a more global constraint which is less susceptible to noise.

5 Conclusion and Discussion

We design and implement a solution for reconstructing objects with curved surfaces. This is particularly applicable to objects whose cross-sections are conics and which can be described by straight homogeneous generalized cylinders. This includes a large set of man-made curved objects. The recovered curved surface also provides a depth map and thus surface interpolation is no longer necessary. When the cross-section of a curved object is not a conic, the terminator of the object can provide information for interpolating the cross-section of other parts of the object.

Other sources of information may also give additional constraints, e.g. specularity, surface markings, and surface texture can be used to obtain the fifth parameter required. By matching surface markings or surface texture, we can deduce the range data of points on a curved surface. When these points are coplanar to the conic being reconstructed, they provide the fifth constraint.

A more global approach for curved-surface reconstruction is to fit one curved surface to all the limbs belonging to the same surface. This will reduce some of the noise problems we encountered. A quadric surface, in general, requires nine parameters. To find a closed form of a quadric surface in terms of the tangent requirements from the limbs is open for research.

References

[Arnold 1981] R. D. Arnold. *Automated Stereo Perception*. PhD. thesis, Stanford University, March, 1981.

[Binford 1987] T. O. Binford. Generic surface interpretation: observability model. In *International Symposium on Robotics Research*, August, 1987.

[Brady 1983] M. Brady and A. Yuille. *An extremum principle for shape from contour*, Technical Report, MIT AI Lab., MIT-AIM 711, 1983.

[Lim 1987] H. S. Lim. *Stereo Vision: Structural Correspondence and Curved Surface Reconstruction.* PhD. Thesis, Stanford University, September, 1987.

[Salmon 1904] G. Salmon. *A Treatise on Conic Sections.* Longmans, Green and Co., 1904.

[Shapira 1978] R. Shapira and H. Freeman. Computer description of bodies bounded by quadric surfaces from a set of imperfect projections. *IEEE Transactions on Computers, Vol. 27, No.9*, 841-854, September, 1978.

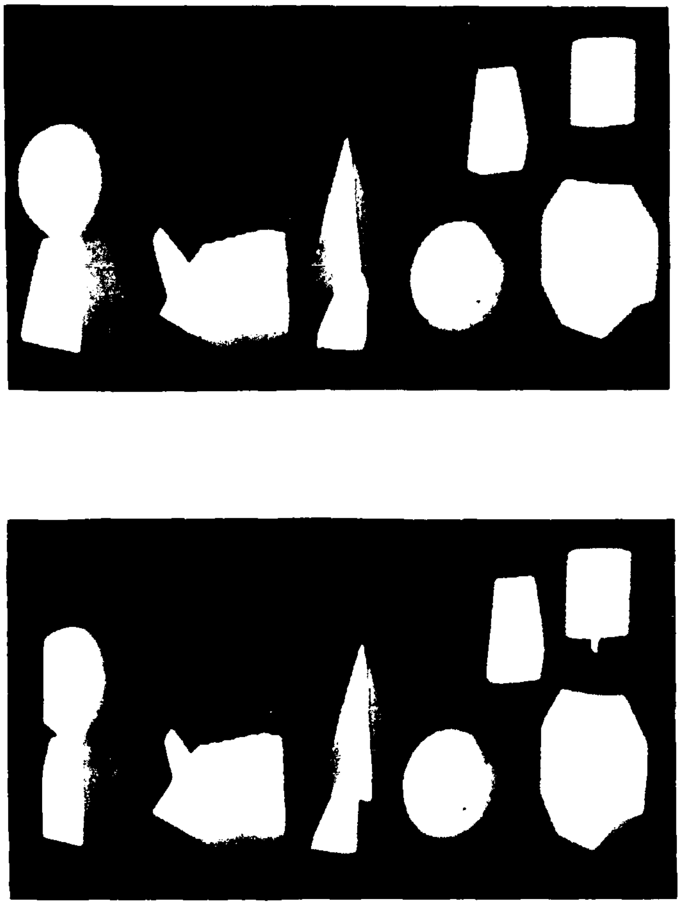


Figure 11: A Pair of Gray-scale Images (Set 2)

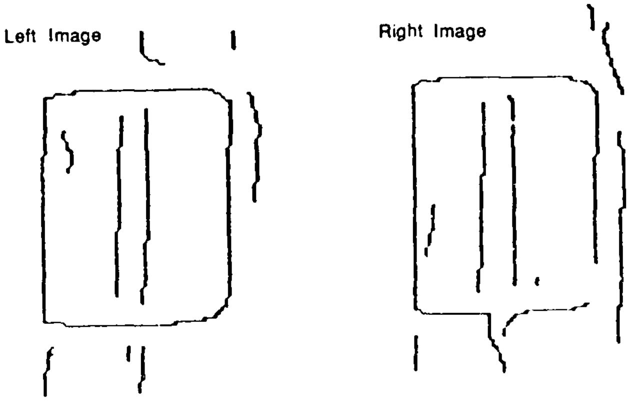


Figure 12: A Pair of Curved Surfaces (Cylinder)

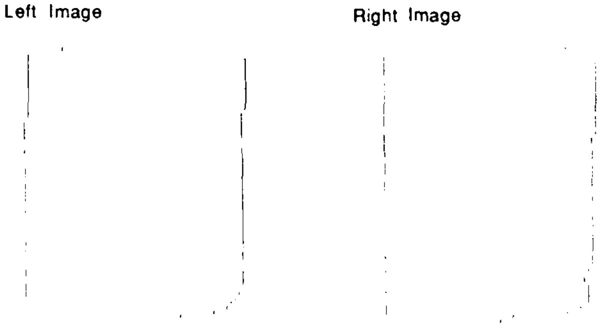


Figure 13: Segmented Curved Surfaces (Cylinder)

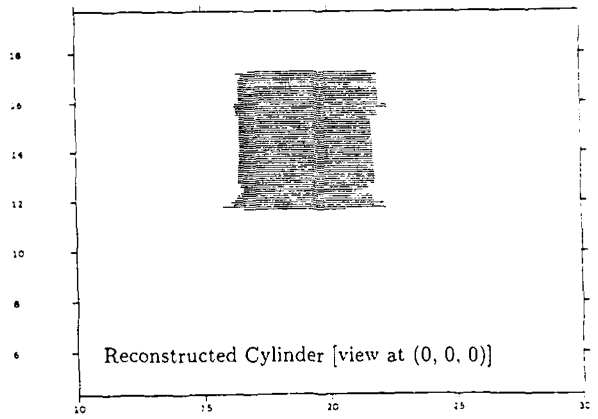


Figure 14: Reconstructed Curved Surface (Cylinder)

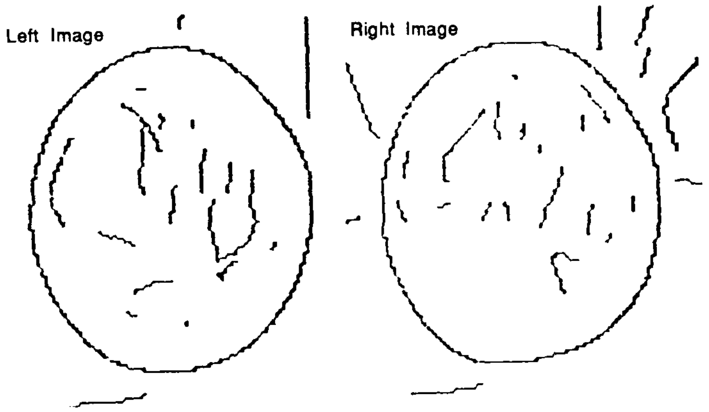


Figure 16: A Pair of Curved Surfaces (Sphere)

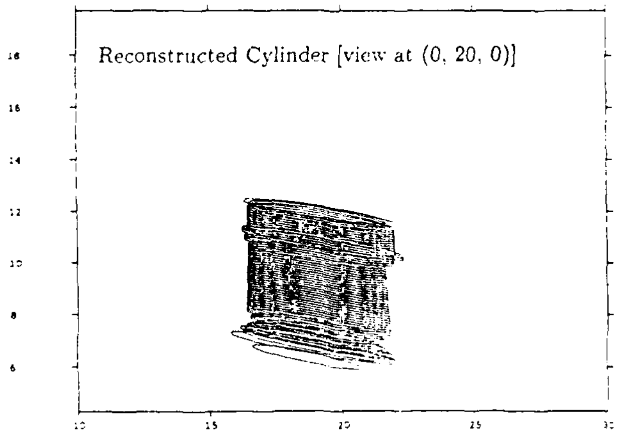


Figure 15: Reconstructed Curved Surface from a Different View (Cylinder)

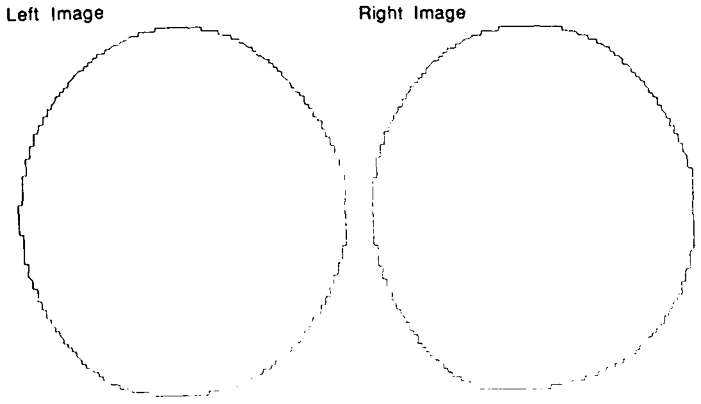


Figure 17: Segmented Curved Surfaces (Sphere)

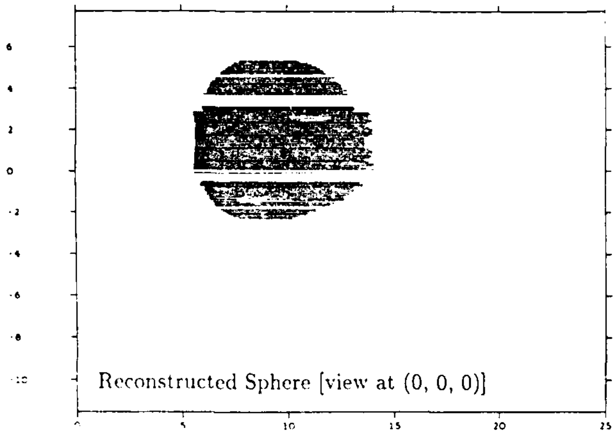


Figure 18: Reconstructed Curved Surface (Sphere)

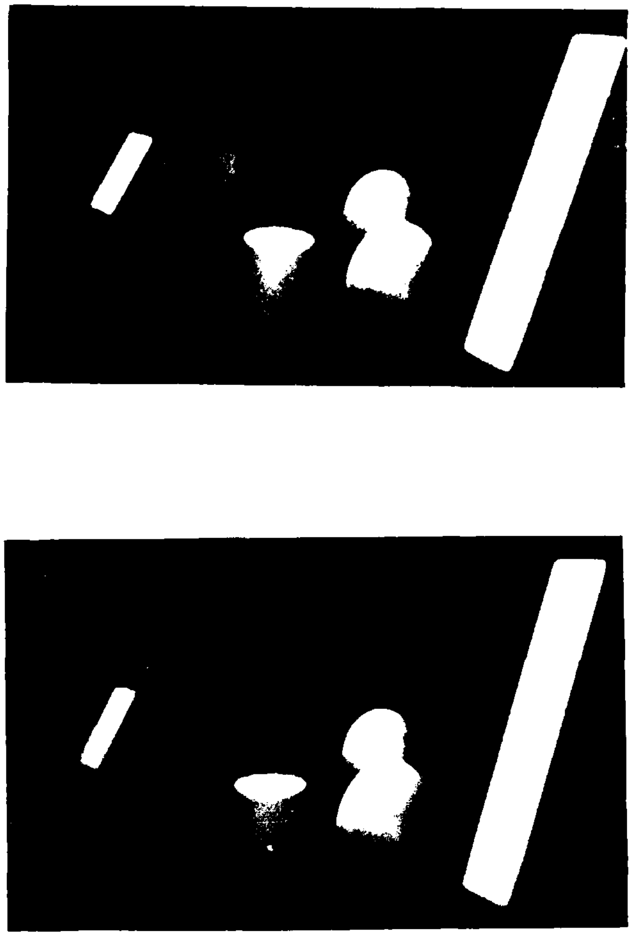


Figure 20: A Pair of Gray-scale Images (Set 1)

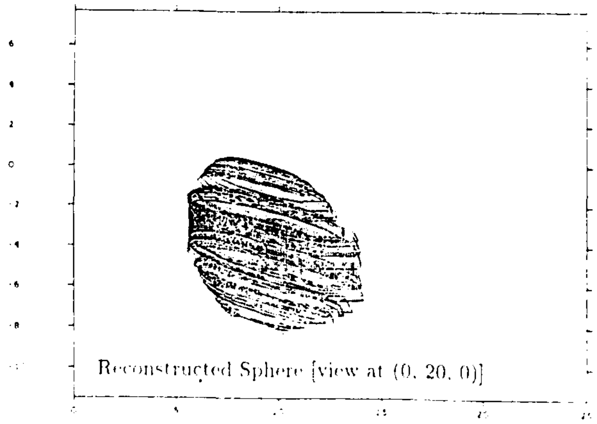


Figure 19: Reconstructed Curved Surface from a Different View (Sphere)

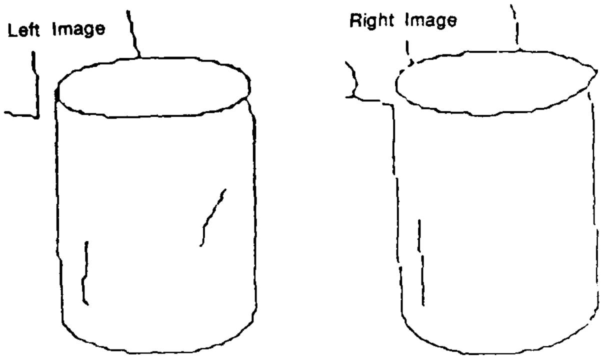


Figure 21: A Pair of Curved Surfaces (Cylinder with Terminator)

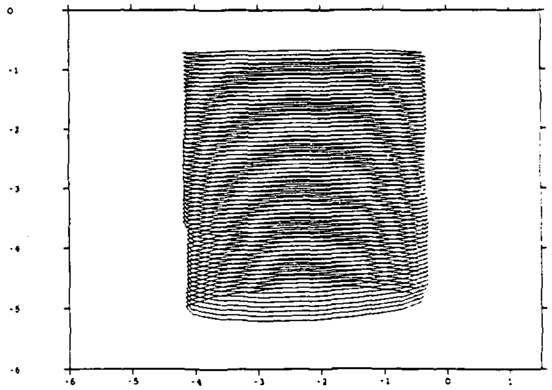


Figure 23: Reconstructed Curved Surface from a Different View (Cylinder with Terminator)

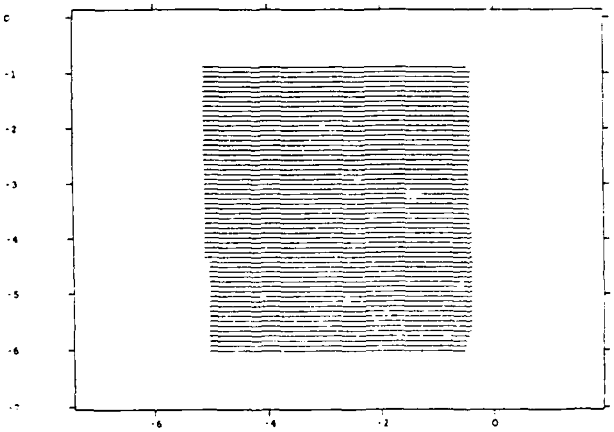


Figure 22: Reconstructed Curved Surface (Cylinder with Terminator)

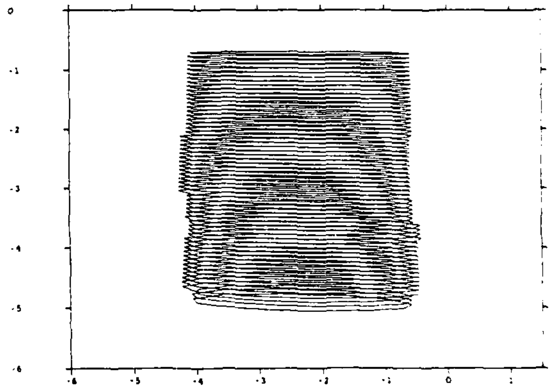


Figure 24: Reconstructed Curved Surface with Extremum Constraint (Cylinder with Terminator)

Geometric Camera Calibration using Systems of Linear Equations*

Keith D. Gremban† Martin Marietta Corporation

Charles E. Thorpe Carnegie Mellon University

Takeo Kanade Carnegie Mellon University

Abstract

Geometric camera calibration is the process of determining a mapping between points in world coordinates and the corresponding image locations of the points. In previous methods, calibration typically involved the iterative solution to a system of non-linear equations. We present a method for performing camera calibration that provides a complete, accurate solution, using only linear systems of equations. By using two calibration planes, a line-of-sight vector is defined for each pixel in the image. The effective focal point of a camera can be obtained by solving the system that defines the intersection point of the line-of-sight vectors. Once the focal point has been determined, a complete camera model can be obtained with a straightforward least squares procedure. This method of geometric camera calibration has the advantages of being accurate, efficient, and practical for a wide variety of applications.

1 Introduction

Many problems in computer vision and graphics require mapping points in space to corresponding points in an image. In computer graphics, for example, an object model is defined with respect to a world coordinate system. To generate an image, the points that lie on the visible surfaces of the object must be mapped onto the image plane; that is, 3d world points must be mapped onto 2d image points. In computer vision, the image locations of points on an object can be used to infer three-dimensional properties of the object; in this case, 2d image points must be mapped back onto the original 3d world points. In both cases, the mapping between 3d world coordinates and 2d image coordinates must be known. Geometric camera calibration is the process of determining the 2d–3d mapping between a camera and a world coordinate system.

We decompose the general problem of geometric camera calibration into two subproblems:

- The projection problem: given the location of a point in space, predict its location in the image; that is, *project* the point into the image.
- The back-projection problem: given a pixel in the image, compute the *line-of-sight* vector through the pixel; that is, *back-project* the pixel into the world.

A complete solution to the camera calibration problem entails deriving a model for the camera geometry that permits the solution of both the projection and the back-projection problems. For many applications, a complete solution is necessary. Some examples from the domain of mobile robots will help illustrate the problems.

In the CMU Navlab project [6], a robot vehicle follows roads using data from a color TV camera. In each image, the road is extracted, and the centerline and direction of the road computed in image coordinates. These parameters are then back-projected into vehicle coordinates and used to plot a course for the vehicle that stays within the road boundaries.

Turk, et al [8] describe a similar road following technique for the Autonomous Land Vehicle (ALV). Rather than parameterizing the road in terms of centerline and direction, they describe the road boundaries as a sequence of points. The line-of-sight vectors for each of the boundary points are computed by back-projection. The intersections of the line-of-sight vectors with the ground plane yield the points in the world between which the robot must steer to stay on the road. In addition, the ALV needs to know the predicted position of the road in each image. This prediction is obtained by projecting the location of the road into each image, based on the position of the road in the previous image, and the motion of the vehicle between images.

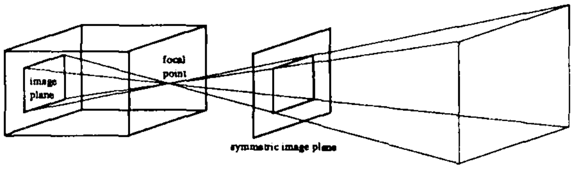


Figure 1: The Pinhole Camera Model

The simplest model for camera geometry is the pinhole, or perspective model. See figure 1. Light rays from in front of the camera converge at the pinhole and are projected onto the image plane at the back of the camera. To avoid dealing with a reversed image, the image plane is often considered to lie in front of the camera, at the same distance from the pinhole. The distance from the focal point to the image plane is the focal length.

A perfect lens can be modeled as a pinhole. No lens is perfect, of course, so part of the problem of geometric camera calibration is correcting for lens distortions. The most accurate and conceptually simple method of camera calibration would be to measure calibration parameters at each pixel in the image. For example, at each pixel measure the line-of-sight vector. This would produce a gigantic lookup table. Then, given a pixel in an image, simple indexing would yield the line-of-sight vector that solves the back-projection problem. To solve the projection problem, the table would be searched to find the line-of-sight vector that passes nearest the point in question.

A lookup table of calibration data for each pixel would be prohibitively expensive. The obvious compromise is to sample the image, and interpolate between data points. If the error in interpolation is less than the measurement error, no accuracy is lost. Most approaches to geometric camera calibration involve sampling the image, and solving for the parameters of the interpolation functions. The obvious differences between approaches are in the form of the interpolation functions, and the mathematical techniques used to solve for the parameters. The main intent of most calibration work has been the solution of the back-projection problem. The projection prob-

*This material is based upon work supported by the Defense Advanced Research Projects Agency and the Army Engineering Topographic Laboratories under contracts DACA76-84-C-0005 and DACA76-85-C-0003, and by the National Science Foundation under grants DCR-8419990 and DCR-8604199. Any opinions, findings, conclusions, or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the Defense Advanced Research Projects Agency, the Army Engineering Topographic Laboratories, the National Science Foundation, or the U.S. Government.

†*author's current address: Computer Science Department, Carnegie Mellon University,* Pittsburgh, PA 15213

lem has occasionally been overlooked. The following paragraphs briefly describe past work.

- Sobel [5] introduced a method for calibration that involved the solution of a large system of non-linear equations. In addition to solving for the intrinsic camera parameters, his method also solved for extrinsic camera parameters, such as camera pan and tilt. Sobel used the basic pinhole model, and solved the system using a non-linear optimization method. He did not model lens distortions, and the system depended on the user to provide initial parameters for the optimization technique.

  Tsai [7] improved on the general non-linear approach in several ways. He modeled distortions globally using fourth order functions, and presented a method for computing good initial parameters for the optimization technique. Tsai's model of lens distortions assumes that the distortions are radially symmetric.

- Yakimovsky and Cunningham [9] presented a calibration technique that also used a pinhole model for the camera. They treated some combinations of parameters as single variables in order to formulate the problem as a system of linear equations. However, in this formulation, the variables are not completely linearly independent, yet are treated as such. No lens distortions are modeled with this approach.

- Martins, Birk, and Kelley [3] reported a calibration technique that does not utilize an explicit camera model. Their two-plane calibration method consisted of measuring the calibration data for various pixels across the image. The data for other pixels is computed by interpolation. The back-projection problem is solved by computing the vector that passes through the interpolated points on each calibration plane. The interpolation can be either local or global. The two-plane method solves only the back-projection problem.

  Isaguirre, Pu, and Summers [2] extended the two-plane method to include calibration as a function of the position and orientation of the camera. They used an iterative approach based on Kalman filters to obtain the solution.

Our goal in camera calibration was to develop a single, basic calibration procedure to solve both the projection and back-projection problems for a variety of applications. Consequently, the desired procedure had to be conceptually straightforward, easily extended to obtain various degrees of accuracy, and computationally efficient. To meet these requirements, we chose to begin with the two-plane method of Martins, Birk, and Kelley. Section 2 discusses the two-plane method and the solution to the back-projection problem. This method can be made arbitrarily accurate; the only problem is that it fails to solve the projection problem. In section 3 we present a method for solving the projection problem that utilizes the calibration data from the two-plane method. The solution to the projection problem is a simple application of analytic geometry, and is completely formulated with systems of linear equations.

The calibration method presented in this paper has been implemented and tested in the Calibrated Imaging Laboratory at CMU [4]. Results are presented in section 4 that demonstrate the accuracy of this method.

2 The Solution to the Back-Projection Problem

Martins, Birk, and Kelly [3] first formally presented the two-plane calibration technique for solving the back-projection problem. This technique has the advantage that it provides exactly the information needed—the ray in space that defines the line of sight of a given pixel—without any explicit camera model.

Figure 2 illustrates the concept of two-plane calibration. Let P_1 and P_2 denote the calibration planes. Assume that the 3d locations of the calibration points on each plane are measured. An image of each plane is acquired, and the image location of each of the calibration points is extracted. Let the calibration points be denoted p_{ij}, and the corresponding image locations be denoted q_{ij}, where $i = 1, 2$ is the plane, and $j = 1, 2, \dots, n$ is the point index. Thus, the image of p_{ij} is q_{ij}.

Let ρ and γ denote the row and column coordinates, respectively, of an image. Then, given a point $v = [\ \rho \quad \gamma\ ]^t$ in the image, the line-of-sight vector for v can be computed as follows. First, use the points p_{1j} and q_{1j} to interpolate the location of v on the first calibration plane, P_1. Call this point u_1. Then, interpolate to find the location of v on the second calibration plane. Call this point u_2. The pixel line-of-sight vector then has direction $u_1 - u_2$ and passes through the point u_1.

Various types of interpolation can be used, with different degrees of accuracy. Martins, et al report three types of interpolation: linear, quadratic, and linear spline. The two-plane method has the potential for being the most accurate of any calibration method for the solution of the back-projection problem. At the limit, this technique consists of measuring the line-of-sight vectors for each pixel in the image. As will be seen in Section 4, the number of calibration points used has a strong influence on the accuracy of the calibration.

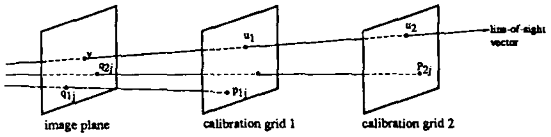


Figure 2: Two-Plane Calibration

2.1 Global Interpolation

One approach to interpolating the calibration data is to globally fit an interpolation function to the data. This function is then used for any pixel across the entire image. Global interpolation has the effect of averaging errors over all the pixels so that the resultant line-of-sight vector is exact for no pixel, but is close for all pixels. This has the advantage of reducing the sensitivity to errors or noise in measurements. On the other hand, the form of the interpolation function is an *a priori* assumption about the lens distortions, and may or may not be appropriate.

2.1.1 Linear Interpolation

Let $p_{ij} = [\ x \quad y \quad z\ ]^t$, and $q_{ij} = [\ \rho \quad \gamma \quad 1\ ]^t$. Then a linear transformation between p and q is given by

$$p_{ij} = A_i q_{ij}$$

where A_i is a 3x3 matrix. Given n measurements on each plane, we can then form the system

$$[\ p_{i1} \quad p_{i2} \quad \dots \quad p_{in}\ ] = A_i [\ q_{i1} \quad q_{i2} \quad \dots \quad q_{in}\ ]$$

or,

$$P_i = A_i Q_i$$

This system can be solved in the least squares sense by using the matrix pseudoinverse (also called the generalized matrix inverse) [1]:

$$A_i = [Q_i^t Q_i]^{-1} Q_i^t P_i$$

Given a pixel v in the image, the direction of the line-of-sight vector through v is given by $u_1 - u_2$ where $u_1 = A_1 v$, and $u_2 = A_2 v$.

2.1.2 Quadratic Interpolation

Quadratic interpolation is similar to linear, except that second-order terms are used in the parameterization, and the matrix A is 6x6. We represent a point in space by $p = [\ x \quad y \quad z\ ]^t$, but we represent image locations by $q = [\ \rho \quad \gamma \quad \rho^2 \quad \gamma^2 \quad \rho\gamma \quad 1\ ]^t$. With these modifications, the formulation is otherwise identical. Martins, et al report that quadratic interpolation was more accurate than linear.

2.2 Local Interpolation

With no *a priori* knowledge about the lens distortions, global interpolation may be inappropriate. A better approach may be to model the distortions locally. If the calibration data is dense enough, the interpolation can be very accurate. In the paragraphs below, we discuss a technique called *linear spline interpolation*, which uses a linear function to perform interpolation over each local region.

Conceptually, this technique of interpolation consists of tesselating each calibration grid with triangles, and performing linear interpolation within each triangle. The calibration points form the vertices of the triangles. A plane is defined uniquely by three points, so no errors are introduced at the vertices. This is not the case for global interpolation techniques, in which errors are averaged over all points, including calibration points. Martins, et al achieved their best accuracy using this form of interpolation. In section 4, we report experiments which confirm this result.

In our current implementation, the grid is not tesselated in advance. Instead, for any point v in the image, each calibration grid is searched to find the three closest calibration points. The linear interpolation matrices A_i are computed using just three points each. The line-of-sight vector is then computed as in Section 2.1.1. Figure 3 illustrates the procedure.

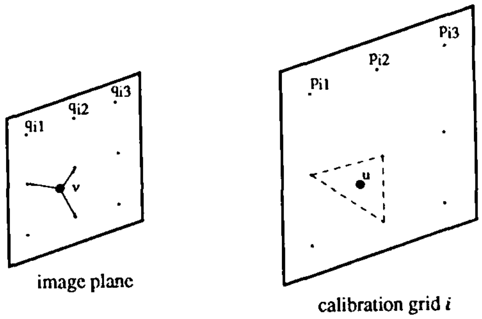


Figure 3: Linear Spline Interpolation

3 Linear Solution to the Projection Problem

The two-plane method for solution to the back-projection problem did not utilize an explicit camera model. In order to use the two-plane data to obtain a solution to the projection problem, it is necessary to have a camera model to formulate the equations. We use a model similar to that of Yakimovsky and Cunningham [9].

In figure 4 we begin with a pinhole model and define the following vectors and points:

- $P = [\ p_x \quad p_y \quad p_z\ ]' ==$ the vector from the origin to a point in space.
- $F = [\ f_x \quad f_y \quad f_z\ ]' ==$ the vector from the origin to the camera focal point.
- $R = [\ r_x \quad r_y \quad r_z\ ]' ==$ a vector that points along the direction of increasing row number. R represents the displacement vector from one pixel to the next in the row direction. The magnitude of R is the row scale factor.
- $C = [\ c_x \quad c_y \quad c_z\ ]' ==$ a vector that points along the direction of increasing column number. C represents the displacement vector from one pixel to the next in the column direction. The magnitude of C is the column scale factor.
- $[\ \rho_p \quad \gamma_p\ ] ==$ the *piercing point* of the image, or the point where the optical axis pierces the image plane.

The vectors R and C define the orientation and scale of the image plane. Columns in the image plane are parallel to R, while rows are parallel to C.

The projection, $[\ \rho \quad \gamma\ ]$, of a point P onto the image plane can be computed by taking the dot product of the vector from the focal point to P, and adding the offset to the piercing point. For example, consider computing the row coordinate, ρ, of the projection. The vector $P - F$ is the vector from the focal point that passes through P. Every point along this vector will have the same location in the image. Let V be a normalized vector along $P - F$, that is, let

$$V = \frac{P - F}{||P - F||} \qquad (1)$$

where $||\cdot||$ denotes the length of a vector. Let $\odot$ represent the usual vector dot product. Then $V \odot R$ represents the projection of V onto R measured in row units. Adding the row coordinate of the piercing point translates $V \odot R$ into image coordinates.

Therefore, the image location, $[\ \rho \quad \gamma\ ]$, of a point P can be computed using the equations:

$$\rho = V \odot R + \rho_p \qquad (2)$$

$$\gamma = V \odot C + \gamma_p \qquad (3)$$

If F, R, C, ρ_p, and γ_p are all unknown, then the resulting system is non-linear. However, the two-plane formulation of the back-projection problem yields the information needed to make solving for the focal point location a linear problem.

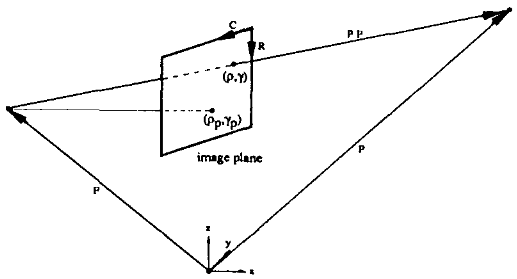


Figure 4: A Linear Model of Camera Geometry

3.1 Focal Point Solution

In a pinhole camera, all incoming light rays pass through the focal point. Since a lens is not a perfect pinhole, we instead refer to an *effective focal point*, which is the point that is closest to all the rays. From the two-plane method for the back-projection problem, one can compute a bundle of rays that pass through the lens. The next step is to find the point in space that minimizes the distance to all the rays.

The equation for the squared distance, d^2, from a point $P = [\ x \quad y \quad z\ ]'$ to the line through $P_1 = [\ x_1 \quad y_1 \quad z_1\ ]'$ in direction $[\ a \quad b \quad c\ ]'$ (where $a^2 + b^2 + c^2 = 1$) is:

$$d^2 = \begin{vmatrix} y - y_1 & z - z_1 \\ b & c \end{vmatrix}^2 + \begin{vmatrix} z - z_1 & x - x_1 \\ c & a \end{vmatrix}^2 + \begin{vmatrix} x - x_1 & y - y_1 \\ a & b \end{vmatrix}^2$$

Expansion of terms yields:

$$\begin{aligned} d^2 &= x^2(b^2 + c^2) + y^2(a^2 + c^2) + z^2(a^2 + b^2) \\ &\quad - 2xyab - 2xzac - 2yzbc \\ &\quad + 2x(bk_3 - ck_2) + 2y(ck_1 - ak_3) + 2z(ak_2 - bk_1) \\ &\quad + k_1^2 + k_2^2 + k_3^2 \end{aligned}$$

where:

$$\begin{aligned} k_1 &= z_1 b - y_1 c \\ k_2 &= x_1 c - z_1 a \\ k_3 &= y_1 a - x_1 b \end{aligned}$$

To find the effective focal point, we need to minimize $D = \sum d^2$. Differentiating D with respect to x, y, and z yields:

$$
\begin{aligned}
\partial D/\partial x &= \sum 2x(b^2+c^2) - \sum 2yab - \sum 2zac + \sum 2(bk_3 - ck_2) \\
\partial D/\partial y &= \sum 2y(a^2+c^2) - \sum 2xab - \sum 2zbc + \sum 2(ck_1 - ak_3) \\
\partial D/\partial z &= \sum 2z(a^2+b^2) - \sum 2xac - \sum 2ybc + \sum 2(ak_2 - bk_1)
\end{aligned}
$$

The sums are taken over all the line-of-sight vectors (a, b, c, k_1, k_2, k_3 are functions of the vectors).

Now, by setting the derivatives of D to zero to find the minima, and putting the equations in matrix form, we obtain:

$$h = Af$$

where:

$$h = \begin{bmatrix} \sum(ck_2 - bk_3) \\ \sum(ak_3 - ck_1) \\ \sum(bk_1 - ak_2) \end{bmatrix}$$

$$A = \begin{bmatrix} \sum(b^2+c^2) & -\sum ab & -\sum ac \\ -\sum ab & \sum(a^2+c^2) & -\sum bc \\ -\sum ac & -\sum bc & \sum(a^2+b^2) \end{bmatrix}$$

$$f = \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

So the solution we seek, f, the effective focal point of the camera, is simply:

$$f = A^{-1}h$$

3.2 Computation of the Camera Base Vectors

Equations (2) and (3) relate the position of a point in space to a corresponding image location. These equations can be written as a linear system:

$$\begin{bmatrix} \rho & \gamma \end{bmatrix} = \begin{bmatrix} v_x & v_y & v_z & 1.0 \end{bmatrix} \begin{bmatrix} r_x & c_x \\ r_y & c_y \\ r_z & c_z \\ \rho_p & \gamma_p \end{bmatrix}$$

Given N points in space, we have $2N$ equations to solve for the 8 unknowns in R, C, ρ_p, and γ_p:

$$\begin{bmatrix} \rho_1 & \gamma_1 \\ \rho_2 & \gamma_2 \\ \vdots & \vdots \\ \rho_n & \gamma_n \end{bmatrix} = \begin{bmatrix} v_{x1} & v_{y1} & v_{z1} & 1.0 \\ v_{x2} & v_{y2} & v_{z2} & 1.0 \\ \vdots & \vdots & \vdots & \vdots \\ v_{xn} & v_{yn} & v_{zn} & 1.0 \end{bmatrix} \begin{bmatrix} r_x & c_x \\ r_y & c_y \\ r_z & c_z \\ \rho_p & \gamma_p \end{bmatrix}$$

or,

$$B = WX$$

So, using the pseudoinverse to obtain a least squares solution, we have:

$$X = [W^tW]^{-1} WB$$

X contains the values for R, C, ρ_p, and γ_p.

3.3 The Local Projection Problem

In section 2 we presented several ways of modeling camera geometry for the back-projection problem. The *linear interpolation* technique (section 2.1.1), which involved fitting a first-order transformation to all the calibration data, is a global modeling technique. The *linear spline interpolation technique* (section 2.2) is a *local* modeling technique, since at each pixel, only the calibration data in a local region around the pixel is used to compute the interpolation function. The results of Martins, et al [3], and our laboratory results (see section 4) both indicate that a local modeling technique yields superior accuracy.

The solution to the projection problem presented in sections 3.1 and 3.2 is a global modeling technique. All the calibration data is used to compute the model parameters, and the results are used to solve the projection problem for any point in space. In direct analogy to the linear spline technique used in back-projection, a technique can be derived for using local information to improve the accuracy of solution to the projection problem.

Our local projection technique involves finding a linear model for local regions of the image. The technique involves two steps. In the first step, the global solution is used to obtain an estimated image location for a point in space. That estimated image location is used to find the four nearest calibration points on each calibration plane. These points are used to compute a local linear solution to the projection problem. The local linear solutions could also be precomputed, and the global solution would then be used simply to index the correct local solution.

4 Experimental Results

Measurements and tests were conducted within the Calibrated Imaging Laboratory (CIL) at CMU (Shafer [4]). The CIL is a facility that provides a precision imaging capability. The purpose of the CIL is to provide researchers with accurate knowledge about ground truth so that computer vision theories can be tested under controlled scientific conditions. Of particular interest for this study, the CIL provides facilities to accurately measure point locations, and to accurately position and orient cameras.

Position measurement of points in the CIL is performed with the use of theodolites (surveyor's transits), which are basically telescopes with crosshairs for sighting, mounted on accurate pan/tilt mechanisms. Objects to be measured are placed at one end of an optical bench; the theodolites are fixed to the other end, separated by a little more than 1 meter. To measure the position of a point, the crosshairs of each theodolite are placed over the point, and the horizontal and vertical displacements read off. Trigonometric equations then yield the position of the point in a Cartesian coordinate system defined with respect to the theodolites. As currently configured, the theodolites can determine point locations to less than 0.1 mm.

4.1 Test Scenario

The laboratory tests described below were designed to provide answers to the following questions:

1. What accuracies can be expected from off the shelf cameras and lenses?
2. How does increasing the number of calibration points affect the accuracy of calibration?
3. What is the expected accuracy for the projection problem?

Tests were performed using a calibrated grid. The grid consisted of horizontal and vertical lines 1mm in width, spaced 12.7 mm apart. The intersections of the lines on the grid were used as calibration points. A special intersection detector was implemented to extract the intersections from digital images with sub-pixel precision. Each time the grid was moved, new measurements were taken, an image digitized, and the intersection detector applied. The result was a data file in which each calibration point was associated with its 3d position and its image location.

A complete test consisted of data from three different grid locations. Due to the size of the laboratory, focal length of the lens, and depth of field of the lens, the grid was typically placed at distances ranging from 0.50m to 0.56m from the camera. Data from two of the grid locations was used to compute calibration parameters. These parameters were then tested using data from the third grid location. The third grid will often be referred to as the test grid. In each of the tests reported here, the focus of the camera was kept fixed. The camera used was a Sony CCD, model AVC-D1, with the standard 16mm lens.

A total of 300 calibration points were used on each grid. Rather than measure the location of each point individually, the location of each point was computed based on the measured locations of the center point and the four corners. Consequently, the accuracy of the data depended not only on the accuracy of the theodolites, but also upon factors such as the planarity of the grid, and the precision of the grid lines. In preparing for each test, the overall accuracy of the calibration data was estimated. For several points at each grid location, the 3d locations were measured using the theodolites. The measured locations were then compared with the computed locations. Differences of up to 0.2 mm were recorded, with typical differences being between 0.1 and 0.2 mm. The accuracy of the calibration method is limited by the accuracy of the calibration data, so the best accuracy achievable in this scenario is between 0.1 and 0.2 mm.

The effects of density of calibration points on calibration accuracy was tested by varying the number of calibration points used. This was easily implemented by simply skipping over some of the rows and columns in the grid. In each case, the calibration points were uniformly distributed over the image. Data is reported for the following distributions of points: 3x3, 5x7, 7x10, 15x20.

In all the tests reported below, grid 0 refers to the grid location farthest from the camera, while grid 2 refers to the grid location closest to the camera. In all cases, the number of points was varied to compute the calibration parameters, but all 300 calibration points on the test grid were used in testing.

4.2 Back-Projection Results

To test the accuracy of the back-projection problem, the image location of each of the calibration points on the third grid was used to compute a line-of-sight vector. The intersection of this vector with the plane of the test grid was computed, and the distance between the intersection and the actual position was used as the error measure. In the results reported below, the errors are averages taken over all the calibration points.

Table 1 presents the results obtained for the back-projection problem. The first error column contains the results for global linear interpolation. For this method, the density of the calibration grid makes little or no difference to the accuracy of the result. This was expected; since the calibration points are uniformly distributed across the grid, additional points do not provide additional information for a linear fit. The best accuracy is achieved when the test grid is positioned between the other two grids used for calibration.

array	calibration	test	error (mm)	
size	grids	grid	global	local
3x3	0, 1	2	1.921	0.731
	0, 2	1	0.388	0.296
	1, 2	0	0.561	0.317
5x7	0, 1	2	1.854	0.740
	0, 2	1	0.366	0.166
	1, 2	0	0.551	0.235
7x10	0, 1	2	1.810	0.696
	0, 2	1	0.350	0.169
	1, 2	0	0.509	0.185
15x20	0, 1	2	1.813	0.666
	0, 2	1	0.366	0.147
	1, 2	0	0.534	0.201

Table 1: Calibration Accuracy of the Back-Projection Problem

The second error column in table 1 presents the results obtained for back-projection problem using local linear spline interpolation. This time there is a general trend for greater accuracy with more calibration points. This reflects the fact that the linear spline method interpolates over local regions, and can more accurately approximate effects such as barrel distortion. There are instances observable in the table which seem to contradict the general trend; these are most likely due to noise in the measurements or in the process of point extraction. Over a number of trials, the general trend has been consistent.

The results in table 1 agree with the results obtained by Martins, et al. To summarize, the local linear spline interpolation procedure, with as few as 12 calibration points, is more accurate than global linear interpolation. In addition, the use of more calibration points improves the accuracy of the local linear spline method. The accuracies we achieved in our tests were at the level of the accuracies of our measurements.

4.2.1 Projection Results

The accuracy of the projection problem was tested with a procedure similar to that used in the back-projection problem. The 3d location of each calibration point on the test grid was projected into the image plane, and the difference (in pixels) between the projected location and the measured location was used as the error measure.

The test results for the projection problem are reported below in table 2. The first error column gives the error, in pixels, of the accuracy using global interpolation. The average error reported in all cases was less than two pixels, which is good enough for many applications. The results indicate that the standard lenses for our cameras are reasonably good, and can be approximated well with a pinhole model.

The second error column in table 2 reports the errors recorded using the local solution to the projection problem.

A comparison of the two columns in table 2 shows an improvement resulting from using local information. In general, the results from the local solution to the projection problem are a factor of 2 improved over the global solution. The entries followed by a * are examples where the global result was better than the local result; this may be an effect of errors in the measurement process. The general conclusion that can be drawn is that local models of camera geometry provide more accurate results than global models—for simple interpolation functions.

array	calibration	test	error (pixels)	
size	grids	grid	global	local
3x3	0, 1	2	1.58	1.70*
	0, 2	1	0.89	0.65
	1, 2	0	0.89	0.82
5x7	0, 1	2	1.32	1.29
	0, 2	1	0.95	0.36
	1, 2	0	0.98	0.46
7x10	0, 1	2	1.20	1.01
	0, 2	1	0.91	0.34
	1, 2	0	0.88	0.40
15x20	0, 1	2	1.15	1.38*
	0, 2	1	0.92	0.92
	1, 2	0	0.91	0.36

Table 2: Accuracy of the Projection Problem

4.2.2 Conclusions

In section 4.1, we enumerated three questions which were to be answered by the tests reported above. We now proceed to answer each of these questions in turn.

1. *What accuracies can be expected from off the shelf cameras and lenses?*

   Tables 1 and 2 of test results show the accuracy achievable with a standard commercial CCD, using the standard lens supplied with the camera. With a simple global interpolation scheme, accuracies as good as 1 part in 1400 (0.3 mm over 530 mm) can be obtained. With a more sophisticated local linear spline interpolation, the accuracies can be increased to 1 part in 3500.

2. *How does increasing the number of calibration points affect the accuracy of calibration?*

   We have shown that the maximam accuracy for global linear interpolation can be achieved with a small number of calibration points, provided that the points are uniformly distributed over the image. Further increasing the number of calibration points has no effect on the accuracy. With a local linear spline interpolation, adding calibration points clearly improves the accuracy of the back-projection problem, until the limiting accuracy of the calibration data is reached.

3. *What is the expected accuracy for the projection problem?*

   Using either a local or global solution, the projection problem can be solved to within two pixels; results as good as 0.34 pixels were reported. For many applications, solution of the projection problem need not be extremely accurate. In many instances, the projected pixel location is only needed to find the center of a region within which an operation will be performed. For these applications, accuracy of one to two pixels is adequate.

It is important to note that the local interpolation outperformed the global interpolation. While the differences were not great in our tests, the lenses we used were fairly linear. If extremely wide angle lenses are used, the distortions may be large, and the ability to locally interpolate will be much more important.

5 Discussion

We have presented a calibration method that we believe meets many of the requirements of a basic calibration technique that can be used for a variety of applications. Our method is based on the two-plane method of Martins, Birk, and Kelley [3], but is extended to include a solution to the projection problem. We believe that the method presented here has many advantages, described in the following paragraphs:

- Completeness.

  The original two-plane method of calibration only provided a solution to the back-projection problem. While this is sufficient for many applications, a solution to the projection problem is also necessary for applications such as mobile robots. We have extended the two-plane method by providing a solution to the projection problem.

- Accuracy.

  The two-plane calibration method can be made arbitrarily accurate. As reported in section 4, increasing the number of calibration points results in increasing accuracy. If no improvement results from adding more points, then the accuracy of the calibration data must be improved.

  The projection problem exhibits much of the same behavior as the back-projection problem. Of particular interest is the observation that local modeling of camera geometry improves the accuracy of the projection problem, as well as the back-projection problem. The accuracies observed in our tests were typically less than one pixel.

- Simplicity

  The two-plane model is conceptually very straightforward and easy to implement. The use of the line-of-sight vectors to solve for the parameters of a linear camera model arises intuitively from the geometry of the model. The method of solution for the camera model involves solving only linear equations, so no sophisticated optimization techniques are involved.

- Efficiency.

  Solution of either the back-projection or projection problems require only a few matrix multiplies and matrix inversions on small matrices. The operations are guaranteed to produce a unique answer within a fixed time. While a relatively large amount of data must be stored for this calibration method compared to other methods, the total amount is still insignificant.

- Practicality.

  Because the method provides a complete solution to the geometric calibration problem, the method can be used for any application. The accuracy can be arbitrarily increased (or decreased) to meet the requirements for a given application. The only change in the method is to store the data from more calibration points. The mathematics remains the same, and no special equipment is required beyond that needed to obtain precise locations for the calibration points.

In addition to the benefits of the method we presented, some general observations should be made:

- Without the benefit of *a priori* knowledge of the form of lens distortions, local modeling of distortions seems to perform better than global modeling. A global model is an attempt to fit the data into a predetermined form and average the error across the entire image. The accuracy of the interpolation is limited by how well the chosen model reflects reality. Conceivably, a different function could be required for different types of lenses to reflect different models of distortion.

  Modeling distortion locally makes no assumption about the forms of the lens distortions. The local model can be made arbitrarily accurate by simply sampling at more pixels. We have shown that relatively few points are needed to achieve the level of accuracy that the measurement devices provide. Moreover, local modeling is more accurate for solving both the projection and back-projection problems.

- Nearly all of the data reported showed that the best accuracy was obtained when the test grid was placed between the two grids used for calibration. This is a specific instance of the general fact that interpolation is more accurate than extrapolation. In calibrating a real robotic system, the calibration data should ideally be obtained so as to bound the region of interest as much as possible.

Geometric camera calibration may depend on a variety of factors. For example, the focal distance, the aperture setting, presence or absence of a filter, or even the operating temperature of a camera may all affect the calibration parameters. We are currently making measurements and conducting tests to determine the sensitivity of calibration parameters to many of these factors.

The results reported in this paper were obtained using data from the Calibrated Imaging Laboratory at CMU. Our next application of this method will be to calibrate and register three cameras and a laser range finder mounted on the CMU Navlab.

References

[1] A. Ben-Israel, and T. N. E. Greville, *Generalized Inverses: Theory and Applications*, John Wiley & Sons, Inc., New York (1974)

[2] A. Isaguirre, P. Pu, and J. Summers, 'A New Development in Camera Calibration: Calibrating a Pair of Mobile Cameras,' *Proc. IEEE Conference on Robotics and Automation*, pp. 74-79 (1985).

[3] H. A. Martins, J. R. Birk, and R. B. Kelley, 'Camera Models Based on Data from Two Calibration Planes,' *Computer Graphics and Image Processing*, 17:173-180 (1981).

[4] S. A. Shafer, 'The Calibrated Imaging Lab Under Construction at CMU,' *Proc. 1985 DARPA Image Understanding Workshop.*

[5] I. Sobel, 'On Calibrating Computer Controlled Cameras for Perceiving 3D Scenes,' *Artificial Intelligence*, 5:185–198 (1974).

[6] C. E. Thorpe, M. Hebert, T. Kanade, and S. Shafer, 'Vision and Navigation for the Carnegie Mellon Navlab,' *Annual Review of Computer Science*, Volume 2, pp. 521–556 (1987), Annual Reviews, Inc.

[7] R. Y. Tsai, 'An Efficient and Accurate Camera Calibration Technique for 3D Machine Vision,' *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pp. 364–374 (1986).

[8] M. A. Turk, D. G. Morgenthaler, K. D. Gremban, and M. Marra, 'Video Road-Following for the Autonomous Land Vehicle,' *Proc. IEEE Conference on Robotics and Automation*, pp. 273–280 (1987).

[9] Y. Yakimovsky, and R. Cunningham, 'A System for Extracting Three-Dimensional Measurements from a Stereo Pair of TV Cameras,' *Computer Graphics and Image Processing* 7:195–210 (1978).

Relative Orientation

Berthold K.P. Horn

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
THE ARTIFICIAL INTELLIGENCE LABORATORY

Abstract: Before corresponding points in images taken with two cameras can be used to recover distances to objects in a scene, one has to determine the position and orientation of one camera relative to the other. This is the classic photogrammetric problem of *relative orientation*, central to the interpretation of binocular stereo information. Iterative methods for determining relative orientation were developed long ago; without them we would not have most of the topographic maps we do today. Relative orientation is also of importance in the recovery of motion and shape from an image sequence when successive frames are widely separated in time.

Described here is a particularly simple iterative scheme for recovering relative orientation that, unlike existing methods, does not require a good *initial* guess for the baseline and the rotation. The data required is a set of pairs of corresponding rays from the two projection centers to points in the scene. It is well known that at least five pairs of rays are needed. Less appears to be known about the existence of multiple solutions and their interpretation. These issues are discussed here in detail. The unambiguous determination of all of the parameters of relative orientation is not possible when the observed points lie on a *critical surface.*

1. Introduction

The positions of corresponding points in two images can be used to determine the positions of points in the environment, provided that the position and orientation of one camera with respect to the other is known. Given the internal geometry of the camera, including its focal length and the location of the principal point, rays can be constructed by connecting the points in the images to their corresponding projection centers. These rays, when extended, intersect at the point in the scene that gave rise to the image points. This is how binocular stereo data is used to determine the positions of points in the environment after the correspondence problem has been solved.

It is also the method used in motion vision when feature points are tracked and the image displacements that occur in the time between two successive frames are relatively large (see for example [Ullman 1979] and [Tsai & Huang 1984]). The connection between these two problems has not attracted much attention before, nor has the relationship of motion vision to some aspects of photogrammetry (but see [Longuet-Higgins 1981]). It turns out, for example, that the well known motion field equations [Longuet-Higgins & Prazdny 1980, Bruss & Horn 1983] are just the *parallax equations* of photogrammetry [Hallert 1960, Moffit & Mikhail 1980] that occur in the incremental adjustment of relative orientation. Most papers on relative orientation only give the equation for y-parallax, corresponding to the equation for the y-component of the motion field (see for example the first equation in [Gill 1964], equation (1) in [Jochmann 1965], and equation (6) in [Oswal 1967]). Some papers actually give equations for both x- and y-parallax (see for example equation (9) in [Bender 1967]).

In both binocular stereo and large displacement motion vision analysis, it is necessary to first determine the *relative orientation* of one camera with respect to the other. The relative orientation can be found if a sufficiently large set of pairs of corresponding image points have been identified [Thompson 1959b, Thompson 1968, Ghosh 1972, Schwidefsky 1973, Slama et. al. 1980, Moffit & Mikhail 1980, Wolf 1983, Horn 1986].

Let us use the terms *left* and *right* to distinguish the two cameras (in the case of the application to motion vision these will be the camera positions and orientations corresponding to the earlier and the later frames respectively).[1]

Acknowledgements: This paper describes research done at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology. Support for the laboratory's artificial intelligence research is provided in part by the Advanced Research Projects Agency of the Department of Defense under Army contract number DACA76-85-C-0100, and in part by the Advanced Research Projects Agency of the Department of Defense under Office of Naval Research contract number N00014-85-K-0124.

[1] In what follows we use the coordinate system of the right (or later) camera as the reference. One can simply interchange left and right if it happens to be more convenient to use the coordinate system of the left (or earlier) camera.

The ray from the center of projection of the left camera to the center of projection of the right camera is called the *baseline* (see Fig. 1). A coordinate system can be erected at each projection center, with one axis along the optical axis, that is, perpendicular to the image plane. The other two axes are in each case parallel to two convenient orthogonal directions in the image plane (such as the edges of the image, or lines connecting pairs of fiducial marks). The rotation of the left camera coordinate system with respect to the right is called the *orientation*.

Note that we cannot determine the length of the baseline without knowledge about the length of a line in the scene, since the ray directions are unchanged if we scale all of the distances in the scene and the baseline by the same positive scale-factor. This means that we should treat the baseline as a unit vector, and that there are really only five unknowns—three for the rotation and two for the direction of the baseline.[2]

2. Existing Solution Methods

Various empirical procedures have been devised for determining the relative orientation in an analog fashion. Most commonly used are stereoplotters, optical devices that permit viewing of image pairs and superimposed synthetic features called floating marks. Differences in ray direction parallel to the baseline are called *horizontal disparities* (or x-parallaxes), while differences in ray direction orthogonal to the baseline are called *vertical disparities* (or y-parallaxes).[3] Horizontal disparities encode distances to points on the surface and are the quantities sought after in measurement of the underlying topography. There should be no vertical disparities when the device is adjusted to the correct relative orientation, since the rays from the left and right projection center must lie in a plane that contains the baseline if they are to intersect.

The methods used in practice to determine the correct relative orientation depend on successive adjustments to eliminate the vertical disparity at each of five or six carefully chosen points [Sailor 1960, Thompson 1964, Slama et al. 1980, Moffit & Mikhail 1980, Wolf 1974]. In each of the adjustments a single parameter of the relative orientation is varied in order to remove the vertical disparity at one of the points. Which adjustment is made to eliminate the vertical disparity at a particular point depends on the particular method is chosen. In each case, however, one of the adjustments, rather than being guided visually, is made by an amount that is calculated, using the measured values of earlier adjustments. The calculation is based on the assumptions that the surface being viewed can be approximated by a plane, that the baseline is roughly parallel to this plane, and that the optical axes of the two cameras are roughly perpendicular to this plane. The whole process is iterative in nature, since the reduction of vertical disparity at one point by means of an adjustment of a single parameter of the relative orientation disturbs the vertical disparity at the other points. Convergence is usually rapid if a good initial guess is available. It can be slow, however, when the assumptions on which the calculation is based are violated, such as in "accidented" or hilly terrain [Van Der Weele 1959-60]. These methods typically use Euler angles to represent three-dimensional rotations [Korn & Korn 1968] (traditionally denoted by the greek letters κ, ϕ, and ω). Euler angles have a number of short-comings for describing rotations that become particularly noticable when these angles become large.

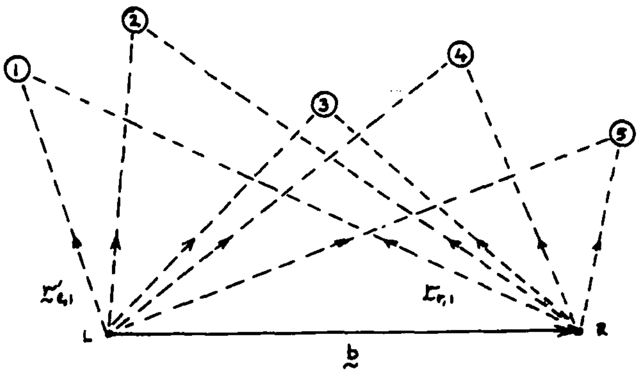


Figure 1. Points in the environment are viewed from two camera positions. The relative orientation is the direction of the baseline **b**, and the rotation R relating the left and right coordinate systems. The directions of rays to at least five scene points must be known in both camera coordinate systems.

There also exist related digital procedures that converge rapidly when a good initial guess of the relative orientation is available, as is usually the case when one is interpreting aerial photography [Slama et al. 1980]. Not all of these methods use Euler angles. Thompson [1959b], for example, uses twice the Gibb's vector [Korn & Korn 1968] to represent rotations. These procedures may fail to converge to the correct solution when the initial guess is far off the mark. In the application to motion vision, approximate translational and rotational components of the motion are

[2] If we treat the baseline as a unit vector, its actual length becomes the unit of length for all other quantities.

[3] This naming convention stems from the observation that, roughly speaking, in the usual viewing arrangement, horizontal disparities correspond to left-right displacements in the image, whereas vertical disparities correspond to up-down displacements.

often not known initially, so a procedure that depends on good initial guesses is not particularly useful.

Normally, the directions of the rays are obtained from points in images generated by projection onto a planar surface. In this case the directions are confined to the field of view as determined by the active area of the image plane and the distance to the center of projection. The field of view is always less than a hemi-sphere, since only points in front of the camera can be imaged.[4] The method described here applies, however, no matter how the directions to points in the scene are determined. There is no restriction on the possible directions of the rays. We do assume, however, that we can tell which of two opposite semi-infinite line-segments the point lies on. If a point lies on the correct line-segment we will say that it lies in *front* of the camera, otherwise it will be considered to be *behind* the camera.

The problem of relative orientation is generally considered solved, and so has received little attention in the photogrammetric literature in recent times [Van Der Weele 1959–60]. In the annual index of *Photogrammetric Engineering*, for example, there is only one reference to the subject in the last ten years [Ghilani 1983] and six in the decade before that. This is very little in comparison to the large number of papers on this subject in the fifties, as well as the sixties, including [Gill 1964], [Sailor 1965], [Jochmann 1965], [Ghosh 1966], [Forrest 1966] and [Oswal 1967].

In this paper we discuss the relationship of relative orientation to the problem of motion vision in the situation where the motion between the exposure of successive frames is relatively large. Also, a new iterative algorithm is described here, as well as a way of dealing with the situation when there is no initial guess available for the rotation or the direction of the baseline. The advantages of the unit quaternion notation for representing rotations are illustrated as well. Finally, we discuss critical surfaces, surface shapes that lead to difficulties in establishing a unique relative orientation.

3. Coplanarity Condition

If the ray from the left camera and the corresponding ray from the right camera are to intersect, they must to lie in a plane that also contains the baseline. Thus, if $\mathbf{b}$ is the vector respresenting the baseline, $\mathbf{r}_r$ the ray from the right projection center to the point in the scene and $\mathbf{r}_l$ the ray from the left projection center to the point in the scene, then the triple product

$$[\mathbf{b}\ \mathbf{r}_l'\ \mathbf{r}_r]$$

equals zero, where $\mathbf{r}_l' = \mathrm{Rot}(\mathbf{r}_l)$ is the left ray rotated into the right camera's coordinate system.[5] This is the *coplanarity condition* (see Fig. 2).

We obtain one such constraint from each pair of rays. There will be an infinite number of solutions for the baseline and the rotation when there are fewer than five pairs of rays, since there are five unknowns and each pair of rays yields only one constraint. Conversely, if there are more than five pairs of rays, the constraints are likely to be inconsistent as the result of small errors in the measurements. In this case, no exact solution of the set of constraint equations exists, and it makes sense instead to minimize the sum of squares of errors in the constraint equations. In practice, one should use more than five pairs of rays in order to reduce the influence of measurement errors [Jochmann 1965, Ghosh 1966]. We shall see later that the added information also allows one to eliminate spurious solutions.

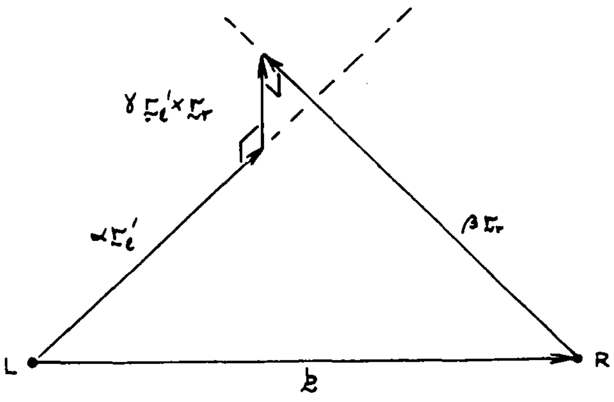


Figure 2. Two rays approach closest where they are intersected by a line perpendicular to both. If there is no measurement error, and the relative orientation has been recovered correctly, then the two rays actually intersect. In this case the two rays and the baseline lie in a common plane.

4. What is the Appropriate Error Term?

The triple product $[\mathbf{b}\ \mathbf{r}_l'\ \mathbf{r}_r]$ is zero when the left and right ray are coplanar with the baseline. It is not immediately apparent, however, that the triple product itself is necessarily the ideal measure of departure from best fit. It is worthwhile exploring the geometry of the two rays more carefully. Consider the points on the rays where they approach each other the closest (see Fig. 2). The line connecting these

[4] The field of view is larger than a hemi-sphere in some fish-eye lenses, where there is significant radial distortion.

[5] The baseline vector $\mathbf{b}$ is here also measured in the coordinate system of the right camera.

points will be perpendicular to both rays, and hence parallel to $(\mathbf{r}_l' \times \mathbf{r}_r)$. As a consequence, we can write

$$\alpha \mathbf{r}_l' + \gamma(\mathbf{r}_l' \times \mathbf{r}_r) = \mathbf{b} + \beta \mathbf{r}_r,$$

where α and β are proportional to the distances along the left and the right ray to the points where they approach most closely, while γ is proportional to the shortest distance between the rays. We can find γ by taking the dot-product of the equality above with $\mathbf{r}_l' \times \mathbf{r}_r$. We obtain

$$\gamma \|\mathbf{r}_l' \times \mathbf{r}_r\|^2 = [\mathbf{b}\ \mathbf{r}_l'\ \mathbf{r}_r].$$

Similarly, taking the dot-products with $\mathbf{r}_r \times (\mathbf{r}_l' \times \mathbf{r}_r)$ and $\mathbf{r}_l' \times (\mathbf{r}_l' \times \mathbf{r}_r)$, we obtain

$$\alpha \|\mathbf{r}_l' \times \mathbf{r}_r\|^2 = (\mathbf{b} \times \mathbf{r}_r) \cdot (\mathbf{r}_l' \times \mathbf{r}_r),$$

$$\beta \|\mathbf{r}_l' \times \mathbf{r}_r\|^2 = (\mathbf{b} \times \mathbf{r}_l') \cdot (\mathbf{r}_l' \times \mathbf{r}_r).$$

The magnitudes of α and β are the distances along the rays to the point of closest approach when $\mathbf{r}_r$ and $\mathbf{r}_l'$ are unit vectors. It turns out, however, that we are more interested here in the signs than in the magnitudes of α and β.

Normally, the points where the two rays approach the closest will be in front of both cameras, that is, both α and β will be positive. If the estimated baseline or rotation is in error, however, then it is possible for one or both of the calculated parameters α and β to be negative. We will use this observation later to distinguish amongst different apparent solutions.

We have shown that the perpendicular distance between the left and the right ray is equal to ratio of the triple product $[\mathbf{b}\ \mathbf{r}_l'\ \mathbf{r}_r]$ to the magnitude squared of $(\mathbf{r}_l' \times \mathbf{r}_r)$. But the measurement errors are in the image, not in the scene. Thus a least-squares procedure should be based on an error *in determining the direction of the rays, not on the distance* of closest approach. To arrive at such a measure, we can look at the angle, θ say, between the projections of the left and right ray into a plane perpendicular to $\mathbf{b}$ (see Fig. 3). This angle will be zero when the vertical disparity is zero, that is, when the left and right rays are coplanar with the baseline.

The projections of $\mathbf{r}_l'$ and $\mathbf{r}_r$ into a plane perpendicular to $\mathbf{b}$ are given by

$$\bar{\mathbf{r}}_l' = \mathbf{r}_l' - (\mathbf{r}_l' \cdot \mathbf{b})\mathbf{b} = (\mathbf{b} \times \mathbf{r}_l') \times \mathbf{b},$$

$$\bar{\mathbf{r}}_r = \mathbf{r}_r - (\mathbf{r}_r \cdot \mathbf{b})\mathbf{b} = (\mathbf{b} \times \mathbf{r}_r) \times \mathbf{b},$$

where we have used the fact that $\mathbf{b}$ is a unit vector. It is easy to show that

$$\|\bar{\mathbf{r}}_l'\| = \|\mathbf{b} \times \mathbf{r}_l'\| \quad \text{and} \quad \|\bar{\mathbf{r}}_r\| = \|\mathbf{b} \times \mathbf{r}_r\|,$$

keeping in mind again that $\mathbf{b}$ is a unit vector, and that $\mathbf{b} \times \mathbf{r}_l'$ and $\mathbf{b} \times \mathbf{r}_r$ are perpendicular to $\mathbf{b}$. The cross-product of the two projected vectors $\bar{\mathbf{r}}_l'$ and $\bar{\mathbf{r}}_r$ will be parallel to the baseline and have magnitude proportional to the sine of the angle between the projected vectors, and so,

$$\sin\theta = \frac{[\mathbf{b}\ \bar{\mathbf{r}}_l'\ \bar{\mathbf{r}}_r]}{\|\bar{\mathbf{r}}_l'\|\,\|\bar{\mathbf{r}}_r\|} = \frac{[\mathbf{b}\ \mathbf{r}_l'\ \mathbf{r}_r]}{\|\mathbf{b} \times \mathbf{r}_l'\|\,\|\mathbf{b} \times \mathbf{r}_r\|},$$

where we have used the fact that $[\mathbf{b}\ \bar{\mathbf{r}}_l'\ \bar{\mathbf{r}}_r] = [\mathbf{b}\ \mathbf{r}_l'\ \mathbf{r}_r]$, something that can easily be verified.

We could use the sine of the angle between the projected vectors directly as a measure of departure from best fit. This is not as good an idea as it may appear at first sight, because of what happens when one of the rays becomes nearly parallel to the baseline. In this case the angle will vary rapidly with small changes in the direction of the ray. Correspondingly, one of the terms in the denominator in the expression for the sine of the angle becomes small. It is better to normalize the expression by multiplying by the lengths of the projected vectors. Then we obtain the area of the parallelogram formed by the projections of the rays into a plane perpendicular to the baseline, namely,

$$\|\bar{\mathbf{r}}_l'\|\,\|\bar{\mathbf{r}}_r\| \sin\theta = [\mathbf{b}\ \bar{\mathbf{r}}_l'\ \bar{\mathbf{r}}_r] = [\mathbf{b}\ \mathbf{r}_l'\ \mathbf{r}_r].$$

This discussion confirms that the triple product itself is a good measure of the departure from best fit. This is convenient, since it makes the least squares analysis reasonably straightforward. If one desires to use a different error measure, one can weight the terms in the sums to follow accordingly.

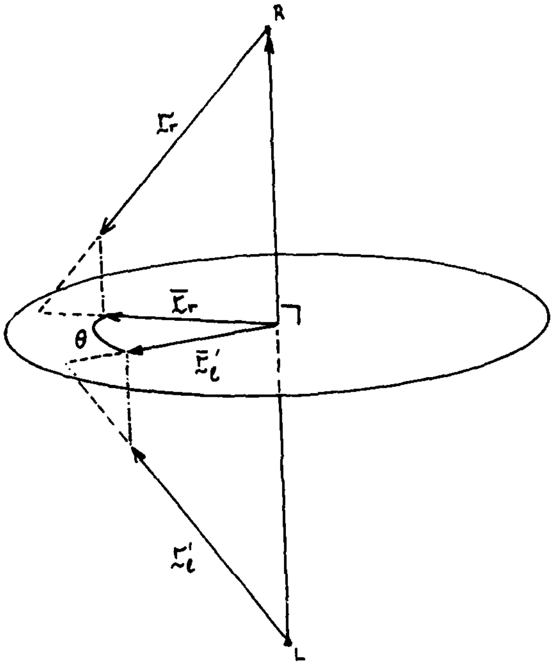


Figure 3. One measure of the departure from the coplanarity condition is the angle θ between the planes formed by the left ray and the baseline and the plane formed by the right ray and the baseline. The angle may be found by projection the two rays onto a plane perpendicular to the baseline.

5. Least Squares Solution for the Baseline

If the rotation is known, it is easy to find the best fit baseline, as we show next. This is useful, despite the fact that we do not usually know the rotation. The reason is that the ability to find the best baseline, given a rotation, reduces the dimensionality of the search space from five to three.

Let $\{\mathbf{r}_{l,i}\}$ and $\{\mathbf{r}_{r,i}\}$, for $i = 1 \ldots n$, be corresponding sets of left and right rays. We wish to minimize

$$E = \sum_{i=1}^{n} [\mathbf{b}\ \mathbf{r}'_{l,i}\ \mathbf{r}_{r,i}]^2 = \sum_{i=1}^{n} \left(\mathbf{b}\cdot(\mathbf{r}'_{l,i}\times\mathbf{r}_{r,i})\right)^2,$$

subject to the condition $\|\mathbf{b}\|^2 = 1$, where $\mathbf{r}'_{l,i}$ is the rotated left ray $\mathbf{r}_{l,i}$, as before. If we let $\mathbf{c}_i = \mathbf{r}'_{l,i} \times \mathbf{r}_{r,i}$, we can rewrite the sum in the simpler form

$$E = \sum_{i=1}^{n} (\mathbf{b}\cdot\mathbf{c}_i)^2 = \mathbf{b}^T \left(\sum_{i=1}^{n} \mathbf{c}_i\mathbf{c}_i^T\right)\mathbf{b},$$

where we have used the equivalence $\mathbf{b}\cdot\mathbf{c}_i = \mathbf{b}^T\mathbf{c}_i$, which depends on the interpretation of column vectors as 3×1 matrices. The term $\mathbf{c}_i\mathbf{c}_i^T$ is a dyadic product, a 3×3 matrix obtained by multiplying a 3×1 matrix by a 1×3 matrix.

The error sum is a quadratic form involving the real symmetric matrix

$$C = \sum_{i=1}^{n} \mathbf{c}_i\mathbf{c}_i^T.$$

The minimum of such a quadratic form is the smallest eigenvalue of the matrix C, attained when $\mathbf{b}$ is the corresponding unit eigenvector (see, for example, the discussion of Rayleigh's quotient in [Korn & Korn 1968]). This can be verified by introducing a Lagrangian multiplier λ and minimizing

$$E' = \mathbf{b}^T C\,\mathbf{b} + \lambda(1 - \mathbf{b}^T\mathbf{b}),$$

subject to the condition $\mathbf{b}^T\mathbf{b} = 1$. Differentiating with respect to $\mathbf{b}$ and setting the result equal to zero yields:

$$C\,\mathbf{b} = \lambda\mathbf{b}.$$

The error corresponding to a particular solution of this equation is found by premultiplying by $\mathbf{b}^T$:

$$E = \mathbf{b}^T C\,\mathbf{b} = \lambda\mathbf{b}^T\mathbf{b} = \lambda.$$

The three eigenvalues of the real symmetric matrix C are non-negative, and can be found in closed form by solving a cubic equation, while each of the corresponding eigenvectors has components that are the solution of three homogeneous equations in three unknowns [Korn & Korn 1968]. If the data are relatively free of measurement error, then the smallest eigenvalue will be much smaller than the other two, and a reasonable approximation to the sought-after result can be obtained by solving for the eigenvector using the assumption that the smallest eigenvalue is actually zero. This way one need not even solve the cubic equation (see also [Horn & Weldon 1988]).

If $\mathbf{b}$ is a unit eigenvector, so is $-\mathbf{b}$. Changing the sense of the baseline does not change the magnitude of the error term $[\mathbf{b}\ \mathbf{r}'_l\ \mathbf{r}_r]$. It does, however, change the signs of α, β and γ. One can decide which sense of the baseline direction is appropriate by determining the signs of α_i and β_i for $i = 1 \ldots n$. Ideally, they should all be positive. The solution for the optimal baseline is not unique unless there are at least two pairs of corresponding rays. The reason is that the eigenvector we are looking for is not uniquely determined if more than one of the eigenvalues is zero, and the matrix has rank less than two if it is the sum of fewer than two dyadic products of independent vectors. This is not a significant restriction, however, since we need at least five pairs of rays to solve for the rotation anyway.

6. Iterative Improvement of Relative Orientation.

If one ignores the orthonormality of the rotation matrix, a set of nine homogeneous linear equations can be obtained by a transformation of the coplanarity conditions that was first described in [Thompson 1959b]. These equations can be solved when eight pairs of corresponding ray directions are known [Longuet-Higgins 1981]. This is not a least-squares method that can make use of redundant measurements, nor can it be applied when fewer than eight points are given. The method is also strongly affected by measurement errors and fails for certain configurations of points [Longuet-Higgins 1984].

No true closed-form solution of the least-squares problem has been found for the general case, where both baseline and rotation are unknown. However, it is possible to determine how the overall error is affected by small changes in the baseline and small changes in the rotation. This allows one to make iterative adjustments to the baseline and the rotation that reduce the sum of squares of errors.

We can represent a small change in the baseline by an infinitesimal quantity $\delta\mathbf{b}$. If this change is to leave the length of the baseline unaltered, then

$$\|\mathbf{b} + \delta\mathbf{b}\|^2 = \|\mathbf{b}\|^2,$$

or

$$\|\mathbf{b}\|^2 + 2\mathbf{b}\cdot\delta\mathbf{b} + \|\delta\mathbf{b}\|^2 = \|\mathbf{b}\|^2.$$

If we ignore quantities of second-order, we obtain

$$\delta\mathbf{b}\cdot\mathbf{b} = 0,$$

that is, $\delta\mathbf{b}$ must be perpendicular to $\mathbf{b}$.

A small change in the rotation can be represented by an infinitesimal rotation vector $\delta\boldsymbol{\omega}$. The direction of this vector is parallel to the axis of rotation, while its magnitude is the angle of rotation. This incremental rotation takes the rotated left ray, $\mathbf{r}'_l$, into

$$\mathbf{r}''_l = \mathbf{r}'_l + (\delta\boldsymbol{\omega}\times\mathbf{r}'_l).$$

This follows from Rodrigues' formula for the rotation of a vector $\mathbf{r}$:

$$\cos\theta\;\mathbf{r}+\sin\theta\,(\boldsymbol{\omega}\times\mathbf{r})+(1-\cos\theta)(\boldsymbol{\omega}\cdot\mathbf{r})\,\boldsymbol{\omega},$$

if we let $\theta=\|\delta\boldsymbol{\omega}\|$, $\boldsymbol{\omega}=\delta\boldsymbol{\omega}/\,\|\delta\boldsymbol{\omega}\|$, and note that $\delta\boldsymbol{\omega}$ is an infinitesimal quantity. Finally then, we see that $[\mathbf{b}\ \mathbf{r}_l'\ \mathbf{r}_r]$ becomes

$$[(\mathbf{b}+\delta\mathbf{b})\ \ (\mathbf{r}_l'+\delta\boldsymbol{\omega}\times\mathbf{r}_l')\ \ \mathbf{r}_r],$$

or,

$$[\mathbf{b}\ \mathbf{r}_l'\ \mathbf{r}_r]\ +\ [\delta\mathbf{b}\ \mathbf{r}_l'\ \mathbf{r}_r]\ +\ [\mathbf{b}\ (\delta\boldsymbol{\omega}\times\mathbf{r}_l')\ \mathbf{r}_r],$$

if we ignore the term $[\delta\mathbf{b}\ (\delta\boldsymbol{\omega}\times\mathbf{r}_l')\ \mathbf{r}_r]$, because it contains the product of two infinitesimal quantities. We can expand two of the triple products in the expression above and obtain

$$[\mathbf{b}\ \mathbf{r}_l'\ \mathbf{r}_r]+(\mathbf{r}_l'\times\mathbf{r}_r)\cdot\delta\mathbf{b}+(\mathbf{r}_r\times\mathbf{b})\cdot(\delta\boldsymbol{\omega}\times\mathbf{r}_l'),$$

or

$$t+\mathbf{c}\cdot\delta\mathbf{b}+\mathbf{d}\cdot\delta\boldsymbol{\omega},$$

for short, where

$$t=[\mathbf{b}\ \mathbf{r}_l'\ \mathbf{r}_r],\quad \mathbf{c}=\mathbf{r}_l'\times\mathbf{r}_r,\quad\text{and}\quad \mathbf{d}=\mathbf{r}_l'\times(\mathbf{r}_r\times\mathbf{b}).$$

Now, we are trying to minimize

$$E=\sum_{i=1}^{n}(t_i+\mathbf{c}_i\cdot\delta\mathbf{b}+\mathbf{d}_i\cdot\delta\boldsymbol{\omega})^2\,,$$

subject to the condition $\mathbf{b}\cdot\delta\mathbf{b}=0$. This constrained minimization problem can be transformed into an equivalent unconstrained form by introduction of a Lagrangian multiplier. Instead of minimizing E itself, we then have to minimize:

$$E'=E+\lambda(\mathbf{b}\cdot\delta\mathbf{b}).$$

Differentiating E' with respect to $\delta\mathbf{b}$, and setting the result equal to zero yields,

$$\sum_{i=1}^{n}(t_i+\mathbf{c}_i\cdot\delta\mathbf{b}+\mathbf{d}_i\cdot\delta\boldsymbol{\omega})\,\mathbf{c}_i+\lambda\mathbf{b}=\mathbf{0}.$$

Before we can proceed, we need to eliminate the unknown Lagrangian multiplier λ. The dot-product of the expression with $\mathbf{b}$ leads to

$$\sum_{i=1}^{n}(t_i+\mathbf{c}_i\cdot\delta\mathbf{b}+\mathbf{d}_i\cdot\delta\boldsymbol{\omega})\,\mathbf{c}_i\cdot\mathbf{b}+\lambda(\mathbf{b}\cdot\mathbf{b})=0,$$

which, since $\mathbf{b}\cdot\mathbf{b}=1$, gives us a value for λ that we can use to compute the term

$$\lambda\mathbf{b}=-\mathbf{b}\sum_{i=1}^{n}(t_i+\mathbf{c}_i\cdot\delta\mathbf{b}+\mathbf{d}_i\cdot\delta\boldsymbol{\omega})\,\mathbf{c}_i\cdot\mathbf{b},$$

or

$$\lambda\mathbf{b}=-(\mathbf{b}\mathbf{b}^T)\sum_{i=1}^{n}(t_i+\mathbf{c}_i\cdot\delta\mathbf{b}+\mathbf{d}_i\cdot\delta\boldsymbol{\omega})\,\mathbf{c}_i.$$

Finally, substituting for $\lambda\mathbf{b}$ in the equation above we obtain

$$B\sum_{i=1}^{n}(t_i+\mathbf{c}_i\cdot\delta\mathbf{b}+\mathbf{d}_i\cdot\delta\boldsymbol{\omega})\,\mathbf{c}_i=\mathbf{0},$$

where

$$B=(I-\mathbf{b}\mathbf{b}^T)$$

is the projection operator that removes components of vectors parallel to the baseline, and I is just the 3×3 identity matrix. We can conclude that the equation above relates quantities in a plane perpendicular to the baseline.

Finally, if we differentiate E' with respect to $\delta\boldsymbol{\omega}$ and set this result also equal to zero, we obtain

$$\sum_{i=1}^{n}(t_i+\mathbf{c}_i\cdot\delta\mathbf{b}+\mathbf{d}_i\cdot\delta\boldsymbol{\omega})\,\mathbf{d}_i=\mathbf{0}.$$

Together, the two vector equations constitute six linear scalar equations in the six unknown components of $\delta\mathbf{b}$ and $\delta\boldsymbol{\omega}$. We can rewrite them in the more compact form:

$$BC\ \delta\mathbf{b}+BF\ \delta\boldsymbol{\omega}=-B\,\overline{\mathbf{c}}$$
$$F^T\ \delta\mathbf{b}\ +\ D\ \delta\boldsymbol{\omega}=-\overline{\mathbf{d}}$$

or

$$\begin{pmatrix}BC & BF\\ F^T & D\end{pmatrix}\begin{pmatrix}\delta\mathbf{b}\\ \delta\boldsymbol{\omega}\end{pmatrix}=-\begin{pmatrix}B\,\overline{\mathbf{c}}\\ \overline{\mathbf{d}}\end{pmatrix}$$

where

$$C=\sum_{i=1}^{n}\mathbf{c}_i\mathbf{c}_i^T,\quad F=\sum_{i=1}^{n}\mathbf{c}_i\mathbf{d}_i^T,\quad\text{and}\quad D=\sum_{i=1}^{n}\mathbf{d}_i\mathbf{d}_i^T,$$

while

$$\overline{\mathbf{c}}=\sum_{i=1}^{n}t_i\mathbf{c}_i\quad\text{and}\quad\overline{\mathbf{d}}=\sum_{i=1}^{n}t_i\mathbf{d}_i.$$

The matrix B is singular, as can be seen by noting that $\mathbf{b}$ is an eigenvector with zero eigenvalue. Thus the first three equations above are not independent. One of them will have to be removed. Fortunately, we also still have to incorporate the condition that $\delta\mathbf{b}$ be perpendicular to $\mathbf{b}$. We can do this by replacing one of the first three equations with the linear equation $\mathbf{b}\cdot\delta\mathbf{b}=0$. For the best numerical accuracy one should eliminate the equation with the smallest coefficients.

The above gives us a way of finding small changes in the baseline and rotation that reduce the overall error sum. This method can be applied iteratively to locate a minimum. Numerical experiments confirm that it converges rapidly when a good initial guess is available. Incremental adjustments cannot be computed if the six-by-six coefficient matrix becomes singular. This will happen when there are fewer than five pairs of rays, and for certain rare configurations of points in the scene (see the discussion of *critical surfaces* later on).

7. Adjusting the Baseline and the Rotation

The iterative adjustment of the baseline is straightforward:

$$\mathbf{b}^{n+1}=\mathbf{b}^n+\delta\mathbf{b}^n,$$

where $\mathbf{b}^n$ is the baseline estimate at the beginning of the n-th iteration, while $\delta\mathbf{b}^n$ is the adjustment computed during the n-th iteration, as discussed in the previous section. If $\delta\mathbf{b}^n$ is not infinitesimal, the result will not be a unit vector. We can, and should, normalize the result by dividing by its magnitude.

7.1. Adjustment of Rotation using Unit Quaternions

Adjusting the rotation is a little harder. Rotations are conveniently represented by unit quaternions [Stuelpnagle 1964, Salamin 1979, Taylor 1982, Horn 1986, Horn 1987a]. The groundwork for the application of the unit quaternion notation in photogrammetry was laid by Thompson [1959a] and Schut [1958–59]. A positive rotation about the axis $\boldsymbol{\omega}$ through an angle θ is represented by the unit quaternion

$$\mathring{q} = \cos(\theta/2) + \sin(\theta/2)\ \boldsymbol{\omega},$$

where $\boldsymbol{\omega}$ is assumed to be a unit vector. Composition of rotations corresponds to multiplication of the corresponding unit quaternions. The rotated version of a vector $\mathbf{r}$ is computed using

$$\mathring{r}' = \mathring{q}\,\mathring{r}\,\mathring{q}^*,$$

where $\mathring{q}^*$ is the conjugate of the quaternion $\mathring{q}$, that is, the quaternion obtained by changing the sign of the vector part. Here, $\mathring{r}$ is a purely imaginary quaternion with vector part $\mathbf{r}$, while $\mathring{r}'$ is a purely imaginary quaternion with vector part $\mathbf{r}'$.

The infinitesimal rotation $\delta\boldsymbol{\omega}$ corresponds to the quaternion

$$\delta\mathring{\omega} = 1 + \delta\boldsymbol{\omega}.$$

We can adjust the rotation $\mathring{q}$ by premultiplying with $\delta\mathring{\omega}$, that is,

$$\mathring{q}^{n+1} = \delta\mathring{\omega}^n\ \mathring{q}^n.$$

If $\delta\boldsymbol{\omega}^n$ is not infinitesimal, $\delta\mathring{\omega}^n$ will not be a unit quaternion, and so the result of the adjustment will not be a unit quaternion either. This undesirable state of affairs can be avoided by using either of the two unit quaternions

$$\delta\mathring{\omega} = \sqrt{1 - \|\delta\boldsymbol{\omega}\|^2} + \delta\boldsymbol{\omega},$$

or

$$\delta\mathring{\omega} = (1 + \delta\boldsymbol{\omega})/\sqrt{1 + \|\delta\boldsymbol{\omega}\|^2}.$$

Alternatively, one can simply normalize the product by dividing by its magnitude.

7.2. Adjustment of Rotation using Orthonormal Matrices

The adjustment of rotation is a little trickier if orthonormal matrices are used to represent rotations. We can write the relationship

$$\mathbf{r}' = \mathbf{r} + (\delta\boldsymbol{\omega} \times \mathbf{r}),$$

in the form

$$\mathbf{r}' = \mathbf{r} + W\,\mathbf{r},$$

where the skew-symmetric matrix W is defined by

$$W = \begin{pmatrix} 0 & -\delta\omega_z & \delta\omega_y \\ \delta\omega_z & 0 & -\delta\omega_x \\ -\delta\omega_y & \delta\omega_x & 0 \end{pmatrix},$$

in terms of the components of rotation vector $\delta\boldsymbol{\omega} = (\delta\omega_x, \delta\omega_y, \delta\omega_z)^T$. Consequently we may write $\mathbf{r}' = Q\,\mathbf{r}$, where $Q = I + W$, or

$$Q = \begin{pmatrix} 1 & -\delta\omega_z & \delta\omega_y \\ \delta\omega_z & 1 & -\delta\omega_x \\ -\delta\omega_y & \delta\omega_x & 1 \end{pmatrix},$$

One could then attempt to adjust the rotation by multiplication of the matrices Q and R as folows:

$$R^{n+1} = Q^n R^n.$$

The problem is that Q is not orthonormal unless $\delta\boldsymbol{\omega}$ is infinitesimal. In practice this means that the rotation matrix will depart more and more from orthonormality as more and more iterative adjustments are made. It is possible to re-normalize this matrix by finding the nearest orthonormal matrix, but this is complicated, involving the determination of the square-root of a symmetric matrix [Horn, Hilden & Negahdaripour 1988]. This is one place where the unit quaternion representation has a distinct advantage: it is trivial to find the nearest unit quaternion to a quaternion that does not have unit magnitude.

To avoid this problem, we should really start with an orthonormal matrix to represent the incremental rotation. We can use either of the unit quaternions

$$\delta\mathring{\omega} = \sqrt{1 - \|\delta\boldsymbol{\omega}\|^2} + \delta\boldsymbol{\omega},$$

or

$$\delta\mathring{\omega} = (1 + \delta\boldsymbol{\omega})/\sqrt{1 + \|\delta\boldsymbol{\omega}\|^2},$$

to construct the corresponding orthonormal matrix

$$Q = \begin{pmatrix} q_0^2 + q_x^2 - q_y^2 - q_z^2 & 2(q_xq_y - q_0q_z) & 2(q_xq_z + q_0q_y) \\ 2(q_yq_x + q_0q_z) & q_0^2 - q_x^2 + q_y^2 - q_z^2 & 2(q_yq_z - q_0q_x) \\ 2(q_zq_x - q_0q_y) & 2(q_zq_y + q_0q_x) & q_0^2 - q_x^2 - q_y^2 + q_z^2 \end{pmatrix},$$

where q_0 is the scalar part of the quaternion $\delta\mathring{\omega}$, while q_x, q_y, q_z are the components of the vector part. Then the adjustment of rotation is accomplished using

$$R^{n+1} = Q^n R^n.$$

Note, however, that the resulting matrices will still tend to depart somewhat from orthonormality due to numerical inaccuracies. This may be a problem if many iterations are required.

8. Inherent Ambiguities

The iterative adjustment described above may arrive at a number of apparently different solutions. Some of these are just different representations of the same solution, while others are related to the correct solution by a simple transformation. First of all, note that $-\mathring{q}$ represents the same rotation as $\mathring{q}$, since

$$(-\mathring{q})\,\mathring{r}\,(-\mathring{q}^*) = \mathring{q}\,\mathring{r}\,\mathring{q}^*.$$

That is, antipodal points on the unit sphere in four dimensions represent the same rotation. We can prevent any confusion by ensuring that the first non-zero component of the resulting unit quaternion is positive, or that the largest component is positive.

Next, note that the triple product, $[\mathbf{b}\ \mathbf{r}'_l\ \mathbf{r}_r]$, changes sign, but not magnitude, when we replace $\mathbf{b}$ with $-\mathbf{b}$. Thus the two possible senses of the baseline yield the same sum of squares of errors. However, changing the sign of $\mathbf{b}$ does change the signs of both α and β. All scene points imaged are in front of the camera, so the distances should all be positive. In the presence of noise, it is possible that some of the distances turn out to be negative, but with reasonable data almost all of them should be positive. This allows us to easily pick the correct sense for the baseline.

Not so obvious is another possibility: Suppose we turn all of the left measurements through π radians about the baseline, in addition to the rotation already determined. That is, replace $\mathbf{r}'_l$ with

$$\mathbf{r}''_l = 2(\mathbf{b}\cdot\mathbf{r}'_l)\mathbf{b} - \mathbf{r}'_l.$$

This follows from Rodrigues' formula for the rotation of a vector $\mathbf{r}$:

$$\cos\theta\ \mathbf{r} + \sin\theta\,(\boldsymbol{\omega}\times\mathbf{r}) + (1-\cos\theta)(\boldsymbol{\omega}\cdot\mathbf{r})\,\boldsymbol{\omega},$$

with $\theta = \pi$ and $\boldsymbol{\omega} = \mathbf{b}$. Then the triple product $[\mathbf{b}\ \mathbf{r}'_l\ \mathbf{r}_r]$ turns into

$$2(\mathbf{b}\cdot\mathbf{r}'_l)[\mathbf{b}\ \mathbf{b}\ \mathbf{r}_r] - [\mathbf{b}\ \mathbf{r}'_l\ \mathbf{r}_r] = -[\mathbf{b}\ \mathbf{r}'_l\ \mathbf{r}_r].$$

This, once again, reverses the sign of the error term, but not its magnitude. Thus the sum of squares of errors is unaltered. The signs of α and β are affected, however, although this time not in as simple a way as when the sense of the baseline was reversed.

If $[\mathbf{b}\ \mathbf{r}'_l\ \mathbf{r}_r] = 0$, we find that exactly one of α and β changes sign. This can be shown as follows: The triple product will be zero when the left and right rays are coplanar with the baseline. In this case we have $\gamma = 0$, and so

$$\alpha\,\mathbf{r}'_l = \mathbf{b} + \beta\,\mathbf{r}_r,$$

Taking the cross-product with $\mathbf{b}$ we obtain

$$\alpha\,(\mathbf{r}'_l\times\mathbf{b}) = \beta\,(\mathbf{r}_r\times\mathbf{b}),$$

If we now replace $\mathbf{r}'_l$ by $\mathbf{r}''_l = 2(\mathbf{b}\cdot\mathbf{r}'_l)\mathbf{b} - \mathbf{r}'_l$, we have for the new distances α' and β' along the rays:

$$-\alpha'\,(\mathbf{r}'_l\times\mathbf{b}) = \beta'\,(\mathbf{r}_r\times\mathbf{b}),$$

We conclude that the product $\alpha'\beta'$ has sign opposite to that of the product $\alpha\beta$. So if α and β are both positive, one of α' or β' must be negative.

In the presence of measurement error the triple product will not be exactly equal to zero. If the rays are nearly coplanar with the baseline, however, we find that one of α and β almost always changes sign. With very poor data, it is possible that both change sign. (Even with totally random ray directions, however, this only happens 27.3% of the time, as determined by Monte Carlo methods). In any case, we can reject a solution in which roughly half the distances are negative. Moreover, we can find the correct solution directly by introducing an additional rotation of π radians about the baseline.

9. Remaining Ambiguity

If we take care of the three apparent two-way ambiguities discussed in the previous section, we find that in practice a unique solution is found, provided that a sufficiently large number of ray pairs are available. That is, the method converges to the unique global minimum from every possibly starting point in parameter space.

Local minima in the sum of squares of errors appear when only a few more than the minimum of five ray pairs are available (as is common in practice). This means that one has to repeat the iteration with different starting values for the rotation in order to locate the global minimum. A starting value for the baseline can be found in each case using the closed-form method described in section 5. To search the parameter space effectively, one needs a way of efficiently sampling the space of rotations. The space of rotations is isomorphic to the unit sphere in four dimensions, with antipodal points identified. The rotation groups of the regular polyhedra provide convenient means of uniformly sampling the space of rotations. The group of rotations of the tetrahedron has 12 elements, that of the hexahedron and the octahedron has 24, and that of the icosahedron and the dodecahedron has 60 (representations of these groups are given in Appendix A for convenience). One can use these as starting values for the rotation. Alternatively, one can just generate a number of randomly placed points on the unit sphere in four dimensions as starting values for the rotation.

When there are only five pairs of rays, the situation is different again. In this case, we have five non-linear equations in five unknowns and so in general expect to find a finite number of exact solutions. That is, it is possible to find baselines and rotations that satisfy the coplanarity conditions exactly and reduce the sum of squares of errors to zero. If we ignore the three types of ambiguities discussed above (incuding the rotation by π radians about the baseline), then there are generally four distinct sets of baselines and rotations that yield an exact solution. Typically only one of these yields positive signs for all of the distances. These are empirical observations; I have not been able to prove that there are generally four solutions that satisfy the coplanarity conditions.

10. Summary of the Algorithm

Consider first the case where we have an initial guess for the rotation. We start by finding the best-fit baseline direction using the closed-form method described in section 5. We determine the correct sense of the baseline by choosing the one that makes most of the signs of the distances positive.

Then we proceed as follows:

- For each pair of corresponding image points, we compute $\mathbf{r}'_{l,i}$, the left ray direction $\mathbf{r}_{l,i}$ rotated into the right camera coordinate system, using the present guess for the rotation.
- We then compute the cross-product $\mathbf{c}_i = \mathbf{r}'_{l,i} \times \mathbf{r}_{r,i}$, the double cross-product $\mathbf{d}_i = \mathbf{r}'_{l,i} \times (\mathbf{r}_{r,i} \times \mathbf{b})$ and the triple-product $t_i = [\mathbf{b}\ \mathbf{r}'_{l,i}\ \mathbf{r}_{r,i}]$.
- We accumulate the dyadic products $\mathbf{c}_i\mathbf{c}_i^T$, $\mathbf{c}_i\mathbf{d}_i^T$ and $\mathbf{d}_i\mathbf{d}_i^T$, as well as the vectors $t_i\mathbf{c}_i$ and $t_i\mathbf{d}_i$. The totals of these quantities over all image point pairs give us the matrices C, F, D and the vectors $\bar{\mathbf{c}}$ and $\bar{\mathbf{d}}$.
- We can now solve for the increment in the baseline $\delta\mathbf{b}$ and the increment in the rotation $\delta\boldsymbol{\omega}$ using the method derived in section 6.
- We adjust the baseline and the rotation using the methods discussed in section 7, and recompute the sum of the squares of the error terms.

The new orientation parameters are then used in the next iteration of the above sequence of steps. As is the case with most iterative procedures, it is hard to decide when to stop. Typically, the total error becomes small after a few iterations and no longer decreases at each step, because of limited accuracy in the arithmetic operations. One can stop the iteration the first time the error increases. Alternatively, one can stop after either a fixed number of iterations or after the error becomes less then some predetermined threshold.

When the decision has been made to stop the iteration, a check of the signs of the distances along the rays is in order. If most of them are negative, the baseline direction should be reversed. If neither sense of the baseline direction yields mostly positive distances, one needs to consider the possibility of a rotation through π radians about the baseline $\mathbf{b}$. If this also yields mixed signs, one is dealing with a local extremum of the error function; something that will only happen if the initial guess is in fact not adequate.

If an initial guess is not available, one proceeds as follows:

- For each rotation in the chosen group of rotations, perform the above iteration to obtain a candidate baseline and rotation.
- Choose the solution that yields the smallest total error.

When there are many pairs of rays, the iterative algorithm will converge to the global minimum error solution from any initial guess for the rotation. There is no need to sample the space of rotations in this case. Also, instead of sampling the space of rotations in a systematic way using a finite group of rotations, one can generate points randomly distributed on the surface of the unit sphere in four-dimensional space. This provides a simpler means of generating initial guesses, although more initial guesses may have to be tried than when a systematic procedure is used to sample the space of rotations.

The method given minimizes the sum of the squares of the triple products

$$[\mathbf{b}\ \mathbf{r}'_l\ \mathbf{r}_r].$$

If desired, one can modify it to use some multiple of the triple product as an error term by weighting the contribution to the overall sum. This can lead to a considerably more complex optimization problem if the weights depend on the unknown baseline and the unknown rotation. This happens, for example, if we try to minimize the sum of squares of the sines of the angles corresponding to vertical disparity:

$$\sin\theta = \frac{[\mathbf{b}\ \mathbf{r}'_l\ \mathbf{r}_r]}{\|\mathbf{b}\times\mathbf{r}'_l\|\,\|\mathbf{b}\times\mathbf{r}_r\|}.$$

If we assume that the weighting factors vary slowly during the iterative process, however, we can to use the current estimates of the baseline and rotation in computing the weighting factors, and not take into account the small variations in the error sum due to changes in the weighting factors. That is, when taking derivatives, the weighting factors are considered constant. This is a good approximation when the parameters vary slowly, as they will when one is close to an extremum.

11. Critical Surfaces

In certain rare cases, relative orientation cannot be recovered fully, even when there are five or more pairs of rays. Normally, each error term varies linearly with distance in parameter space from the location of an extremum, and so the sum of squares of errors varies quadratically. There are situations, however, where the error terms to not vary linearly with distance, but quadratically or higher order, in certain special directions in parameter space. In this case, the sum of squares of errors does not vary quadratically with distance from the extremum, but as a function of the fourth or higher power of this distance. This makes it very difficult to accurately locate the extremum. In an extreme situation, the total error may not vary at all along some curve in parameter space. In this case, the total error is unaffected by a small change in the rotation, as long as this change is accompanied by a corresponding small change in the baseline. There is no localized extremum and consequently the solution for relative orientation is not unique. It turns out that this problem arises only when the observed scene points lie on certain surfaces called *Gefährliche Flächen* or *critical surfaces* [Brandenberger 1947, Hofmann 1949, Zeller 1952, Schwidefsky 1973]. We show next that only points on certain hyperboloids of one sheet and their degenerate forms can lead to this kind of problem (see also [Horn 1987b]).

We could try to find a direction of movement in parameter space ($\delta\mathbf{b}$, $\delta\boldsymbol{\omega}$) that leaves the total error unaffected when given a particular surface. Instead, we will take the critical direction of motion in the parameter space as given, and try to find a surface for which the total error is unchanged.

Let $\mathbf{R}$ be a point on the surface, measured in the right camera coordinate system. Then

$$\beta\,\mathbf{r}_r = \mathbf{R} \quad \text{and} \quad \alpha\,\mathbf{r}_l' = \mathbf{b} + \mathbf{R},$$

for some positive α and β. In the absence of measurement errors,

$$[\mathbf{b}\ \mathbf{r}_l'\ \mathbf{r}_r] = \frac{1}{\alpha\beta}[\mathbf{b}\ (\mathbf{b}+\mathbf{R})\ \mathbf{R}] = 0.$$

We noted earlier that when we change the baseline and the rotation slightly, the triple product $[\mathbf{b}\ \mathbf{r}_l'\ \mathbf{r}_r]$ becomes

$$[(\mathbf{b}+\delta\mathbf{b})\ (\mathbf{r}_l' + \delta\boldsymbol{\omega}\times\mathbf{r}_l')\ \mathbf{r}_r],$$

or, if we ignore higher-order terms,

$$[\mathbf{b}\ \mathbf{r}_l'\ \mathbf{r}_r] + (\mathbf{r}_l'\times\mathbf{r}_r)\cdot\delta\mathbf{b} + (\mathbf{r}_r\times\mathbf{b})\cdot(\delta\boldsymbol{\omega}\times\mathbf{r}_l').$$

The problem we are focusing on here arises when this error term is unchanged for small movement in some direction in the parameter space. That is when

$$(\mathbf{r}_l'\times\mathbf{r}_r)\cdot\delta\mathbf{b} + (\mathbf{r}_r\times\mathbf{b})\cdot(\delta\boldsymbol{\omega}\times\mathbf{r}_l') = 0,$$

for some $\delta\mathbf{b}$ and $\delta\boldsymbol{\omega}$. Introducing the coordinates of the imaged points we obtain:

$$\frac{1}{\alpha\beta}\Big(((\mathbf{b}+\mathbf{R})\times\mathbf{R})\cdot\delta\mathbf{b} + (\mathbf{R}\times\mathbf{b})\cdot\big(\delta\boldsymbol{\omega}\times(\mathbf{b}+\mathbf{R})\big)\Big) = 0,$$

or

$$(\mathbf{R}\times\mathbf{b})\cdot(\delta\boldsymbol{\omega}\times\mathbf{R}) + (\mathbf{R}\times\mathbf{b})\cdot(\delta\boldsymbol{\omega}\times\mathbf{b}) + [\mathbf{b}\ \mathbf{R}\ \delta\mathbf{b}] = 0.$$

If we expand the first of the dot-products of the cross-products, we can write this equation in the form

$$(\mathbf{R}\cdot\mathbf{b})(\delta\boldsymbol{\omega}\cdot\mathbf{R}) - (\mathbf{b}\cdot\delta\boldsymbol{\omega})(\mathbf{R}\cdot\mathbf{R}) + \mathbf{L}\cdot\mathbf{R} = 0,$$

where

$$\mathbf{L} = \boldsymbol{\ell}\times\mathbf{b}, \quad \text{while} \quad \boldsymbol{\ell} = \mathbf{b}\times\delta\boldsymbol{\omega} + \delta\mathbf{b}.$$

The expression on the left-hand side contains a part that is quadratic in $\mathbf{R}$ and a part that is linear. The expression is clearly quadratic in X, Y, and Z, the components of the vector $\mathbf{R} = (X, Y, Z)^T$. Thus a surface leading to the kind of problem described above must be a quadric surface [Korn & Korn 1968].

Note that there is no constant term in the equation of the surface, so $\mathbf{R} = \mathbf{0}$ satisfies the equation. This means that the surface passes through the right projection center. It is easy to verify that $\mathbf{R} = -\mathbf{b}$ satisfies the equation also, which means that the surface passes through the left projection center as well. In fact, the whole baseline (and its extensions), $\mathbf{R} = k\mathbf{b}$, lies in the surface. This means that we must be dealing with a ruled quadric surface. It can consequently not be an ellipsoid or hyperboloid of two sheets, or one of their degenerate forms. The surface must be a hyperboloid of one sheet, or one of its degenerate forms. Additional information about the properties of these surfaces is given in Appendix B, while the degenerate forms are explored in Appendix C.

It should be apparent that this kind of ambiguity is quite rare. This is nevertheless an issue of practical importance, however, since the accuracy of the solution is reduced if the points lie near some critical surface. A textbook case of this occurs in aerial photography of a roughly U-shaped valley taken along a flight line parallel to the axis of the valley from a height above the valley floor approximately equal to the width of the valley. In this case the baseline lies on a circular cylinder that also lies close to the surface on which the imaged points lie. This means that it is close to one of the degenerate forms of the hyperboloid of one sheet (see Appendix C).

Note that hyperboloids of one sheet and their degenerate forms are exactly the surfaces that lead to ambiguity in the case of motion vision. The coordinate systems and symbols have been chosen here to make the correspondence between the two problems more obvious. The relationship between the two situations is nevertheless not quite as transparent as I had thought earlier [Horn 1987b]. In the case of the ambiguity of the motion field, for example, we are dealing with a two-way ambiguity arising from infinitesimal displacements. Here we are dealing with an infinite number of solutions arising from images taken with cameras that have large differences in position and orientation. Also note that the symbol $\delta\boldsymbol{\omega}$ stands for a small change in a finite rotation here, while it refers to a difference in instantaneous rotational velocities in the motion vision case.

12. Conclusions

Methods for recovering the relative orientation of two cameras are of importance in both binocular stereo and motion vision. A new iterative method for finding the relative orientation has been described here. It can be used even when there is no initial guess available for the rotation or the baseline. The new method does not use Euler angles to represent the orientation.

When there are many pairs of corresponding image points, the iterative method finds the global minimum from any starting point in parameter space. Local minima in the sum of squares of errors occur, however, when there are relatively few pairs of corresponding image points available. Method for efficiently locating the global minimum in this case were discussed. When only five pairs of corresponding image points are given, several exact solutions of the coplanarity equations can be found. Typically only one of these yields positive distances to the points in the scene, however. This allows one to pick the correct solution even when there is no initial guess available.

All of these methods fail in the rare case that the scene points lie on a critical surface.

Acknowlegements

I would like to thank W. Eric L. Grimson and Rodney A. Brooks, who made helpful comments on a draft of this paper and Shahriar Negadahripour, who worked on the application of results developed earlier for critical surfaces in motion vision in the case when the step between successive images is relatively large.

References

Bender, L.U. (1967) "Derivation of Parallax Equations," *Photogrammetric Engineering*, Vol. 33, No. 10, pp. 1175-1179, October.

Brandenberger, A. (1947) "Fehlertheorie der äusseren Orientierung von Steilaufnahmen," Ph.D. Thesis, Eidgenössisch Technische Hochschule, Zürich, Switzerland.

Bruss, A.R. & B.K.P. Horn, (1983) "Passive Navigation," *Computer Vision, Graphics, and Image Processing*, Vol. 21, No. 1, January, pp. 3-20.

Brou, P. (1983) "Using the Gaussian Image to Find the Orientation of an Object," *International Journal of Robotics Research*, Vol. 3, No. 4, pp. 89–125.

Forrest, R.B. (1966) "AP-C Plotter Orientation," *Photogrammetric Engineering*, Vol. 32, No. 5, pp. 1024–1027, September.

Ghilani, C.D. (1983) "Numerically Assisted Relative Orientation of the Kern PG-2," *Photogrammetric Engineering and Remote Sensing*, Vol. 49, No. 10, pp. 1457–1459, October.

Gill, C. (1964) "Relative Orientation of Segmented, Panoramic Grid Models on the AP-II," *Photogrammetric Engineering*, Vol. 30, pp. 957–962, month?

Ghosh, S.K. (1966) "Relative Orientation Improvement," *Photogrammetric Engineering*, Vol. 32, No. 3, pp. 410 414, May.

Ghosh, S.K. (1972) *Theory of Stereophotogrammetry*, Ohio University Bookstores, Columbus, OH.

Hallert, B. (1960) *Photogrammetry*, McGraw-Hill, New York, NY.

Hilbert, D. & S. Cohn-Vossen (1953, 1983) *Geometry and the Imagination*, Chelsea Publishing, New York.

Horn, B.K.P. (1986) *Robot Vision*, MIT Press, Cambridge, MA & McGraw-Hill, New York, NY.

Horn, B.K.P. (1987a) "Closed-form Solution of Absolute Orientation using Unit Quaternions," *Journal of the Optical Society A*, Vol. 4, No. 4, pp. 629–642, April.

Horn, B.K.P. (1987b) "Motion Fields are Hardly Ever Ambiguous," *International Journal of Computer Vision*, Vol. 1, No. 3, pp. 263–278, Fall.

Horn, B.K.P. & E.J. Weldon Jr. (1988) "Direct Methods for Recovering Motion," accepted for publication by *International Journal of Computer Vision.*

Horn, B.K.P., H.M. Hilden & S. Negahdaripour (1988) "Closed-form Solution of Absolute Orientation using Orthonormal Matrices," submitted to *Journal of the Optical Society A*.

Hofmann, W. (1949) "Das Problem der 'Gefährlichen Flächen' in Theorie and Praxis," Ph.D. Thesis, Technische Hochschule München, Published in 1953 by Deutsche Geodätische Kommission, München, West Germany.

Jochmann, H. (1965) "Number of Orientation Points," *Photogrammetric Engineering*, Vol. 31. No. 4, pp. 670–679, July.

Korn, G.A. & T.M. Korn (1968) *Mathematical Handbook for Scientists and Engineers*, 2-nd edition, McGraw-Hill, New York, NY.

Longuet-Higgins, H.C. & K. Prazdny (1980) "The Interpretation of a Moving Retinal Image," *Proc. of the Royal Society of London B*, Vol. 208, pp. 385–397.

Longuet-Higgins, H.C. (1981) "A Computer Algorithm for Reconstructing a Scene from Two Projections," *Nature*, Vol. 293, pp. 133–135, September.

Longuet-Higgins, H.C. (1984) "The Reconstruction of a Scene from Two Projections—Configurations that Defeat the Eight-Point Algorithm," *IEEE, Proceedings of the First Conference on Artificial Intelligence Applications*, Denver, Colorado.

Moffit, F. & E. M. Mikhail (1980) *Photogrammetry*, 3-rd edition, Harper & Row, New York, NY.

Oswal, H.L. (1967) "Comparison of Elements of Relative Orientation," *Photogrammetric Engineering*, Vol. 33, No. 3, pp. 335–339, March.

Sailor, S. (1965) "Demonstration Board for Stereoscopic Plotter Orientation," *Photogrammetric Engineering*, Vol. 31, No. 1, pp. 176–179, January.

Salamin, E. (1979) "Application of Quaternions to Computation with Rotations," Unpublished Internal Report, Stanford University, Stanford, CA.

Schut, G.H. (1958–59) "Construction of Orthogonal Matrices and Their Application in Analytical Photogrammetry," *Photogrammetria*, Vol. 15, No. 4, pp. 149–162.

Schwidefsky, K. (1973) *An Outline of Photogrammetry*, (Translated by John Fosberry), 2-nd edition, Pitman & Sons, London, England.

Slama, C.C., C. Theurer & S.W. Henrikson, (eds.) (1980) *Manual of Photogrammetry*, American Society of Photogrammetry, Falls Church, VA.

Stuelpnagle, J.H. (1964) "On the Parametrization of the Three-Dimensional Rotation Group," *SIAM Review*, Vol. 6, No. 4, pp. 422–430, October.

Taylor, R.H. (1982) "Planning and Execution of Straight Line Manipulator Trajectories," in *Robot Motion: Planning and Control*, Brady, M.J., J.M. Hollerbach, T.L. Johnson, T. Lozano-Pérez & M.T. Mason (eds.), MIT Press, Cambridge, MA.

Thompson, E.H. (1959a) "A Method for the Construction of Orthonormal Matrices," *Photogrammetric Record*, Vol. 3, No. 13, pp. 55–59, April.

Thompson, E.H. (1959b) "A Rational Algebraic Formulation of the Problem of Relative Orientation," *Photogrammetric Record*, Vol. 3, No. 14, pp. 152–159, October.

Thompson, E.H. (1964) "A Note on Relative Orientation," *Photogrammetric Record*, Vol. 4, No. 24, pp. 483–488, October.

Thompson, E.H. (1968) "The Projective Theory of Relative Orientation," *Photogrammetria*, Vol. 23, pp. 67–75.

Tsai, R.Y. & T.S. Huang (1984) "Uniqueness and Estimation of Three-Dimensional Motion Parameters of Rigid Objects with Curved Surfaces," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 6, No. 1, pp. 13–27, January.

Ullman, S. (1979) *The Interpretation of Visual Motion*, MIT Press, Cambridge, MA.

Van Der Weele, A.J. (1959–60) "The Relative Orientation of Photographs of Mountainous Terrain," *Photogrammetria*, Vol. 16, No. 2, pp. 161–169.

Wolf, P.R. (1983) *Elements of Photogrammetry*, 2-nd edition, McGraw-Hill, New York, NY.

Zeller, M. (1952) *Textbook of Photogrammetry*, H.K. Lewis & Company, London, England

Image Segmentation and Reflection Analysis Through Color

Gudrun J. Klinker, Steven A. Shafer and Takeo Kanade

Computer Science Department, Carnegie-Mellon University, Pittsburgh PA 15213

Abstract

In this paper, we present an approach to color image understanding that can be used to segment and analyze surfaces with color variations due to highlights and shading. We begin with a theory that relates the reflected light from dielectric materials, such as plastic, to fundamental physical reflection processes, and describes the color of the reflected light as a linear combination of the color of the light due to surface reflection (highlights) and body reflection (object color). This theory is used in an algorithm that separates a color image into two parts: an image of just the highlights, and the original image with the highlights removed. In the past, we have applied this method to hand-segmented images. The current paper shows how to perform automatic segmentation method by applying this theory in stages to identify the object and highlight colors. The result is a combination of segmentation and reflection analysis that is better than traditional heuristic segmentation methods (such as histogram thresholding), and provides important physical information about the surface geometry and material properties at the same time. We also show the importance of modeling the camera properties for this kind of quantitative analysis of color. This line of research can lead to physics-based image segmentation methods that are both more reliable and more useful than traditional segmentation methods.[1]

1. Introduction

One of the key goals of computer vision is to interpret the scene as a collection of shiny and matte surfaces, smooth and rough, interacting with light, shape, and shadow. However, computer vision has not yet been successful at deriving such a description of surface and illumination properties from an image. The key reason for this failure has been a lack of models or descriptions rich enough to relate pixels and pixel-aggregates to scene characteristics. In the past, most work with color images has considered object color to be a constant property of an object, and color variation on an object was attributed to noise[1,2,3]. However, color variation in real scenes depends to a much larger degree on the optical reflection properties of the scene. This variation causes the perception of object color, highlights, shadows and shading[4,5], scene properties that can be determined and used by color vision algorithms.

This paper presents an approach to color image understanding that accounts for color variations due to highlights and shading. We use a reflection model which describes pixel colors as a linear combination of an object color and a highlight color[6]. All color pixels from one object form a planar cluster in the color space. The cluster shape is determined by the object and highlight colors and by the object shape and illumination geometry. We extend our reflection model with a sensor model which accounts for camera properties, such as a limited dynamic range, blooming, gamma-correction, and chromatic aberration. This allows us to apply our algorithms to real images, instead of only to synthesized images.

We have previously reported how this theory can be used to separate color images into two intrinsic images, one showing the scene without highlights, and the other showing only the highlights[7]. In the past, we have applied this method to hand-segmented images. The current paper describes an automatic segmentation method that is based on the extended dichromatic reflection model. Our approach alternates between generating hypotheses about the scene from the image data and verifying whether the hypotheses fit the image. The hypotheses relate object color, shading, highlights and camera limitations to the shapes of color clusters in local image areas. By using this control structure, driven by a physical model of light reflection, we are able to incrementally identify local and global properties of the scene, such as object and illumination colors, and to use them in interpreting pixels in the images. This allows us to adapt the image interpretation process to local scene characteristics and to react differently to color and intensity changes at different places in the image. This method stands in contrast to Gershon's approach[8], in which he begins with a traditional segmentation and follows it with a physics-based post-processing step. His approach suffers from erroneous region boundaries created by the initial segmentation, while our

[1] This material is based upon work supported by the National Science Foundation under Grant DCR-8419990 and by the Defense Advanced Research Projects Agency (DOD), ARPA Order No. 4976, monitored by the Air Force Avionics Laboratory under contract F33615-84-K-1520. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the National Science Foundation, the Defense Advanced Research Projects Agency, or the US Government.

method uses the physical model from the beginning and therefore does not make such unrecoverable mistakes.

The result is a combination of segmentation and reflection analysis that is better than traditional heuristic segmentation methods that base their analysis on intensity or color differences or on a fixed set of user-defined features, such as intensity, hue and saturation. Traditional algorithms also cannot distinguish reliably between different edge types, such as highlight edges, material edges and shading or shadow edges, and they cannot account for camera limitations. Moreover, our method generates important physical information about the scene. This information significantly simplifies our subsequent step to separate color images into two intrinsic reflection images[7], since the segmentation already provides all necessary information about color variation on an object. When combined with methods to interpret the intrinsic images[9, 4], this line of research can lead to physics-based image segmentation methods that are both more reliable and more useful than traditional segmentation methods.

2. The Dichromatic Reflection Model

On its path from a light source to the camera, a light ray is altered in many characteristic ways by the objects in the scene. The camera then encodes the measured light in a (color) pixel. It is the goal of image understanding methods to use properties and relationships between pixels to recover a description of the scene. It is essential to the success of such methods that they understand and model the reflection processes in the scene, as well as the sensing characteristics of the camera. How light is reflected from an object depends on the material of the object. It is common to distinguish between conducting materials, such as metals, and dielectric (non-conducting) materials, such as plastics, paints, papers and ceramics. In the following, we will present a model that describes the reflection processes of dielectric materials.

Most dielectric materials are inhomogenecus. They consist of a medium and some embedded pigments, as shown in Figure 2-1. The pigments selectively absorb light and scatter it by reflection and refraction. When we look at objects that are made out of dielectric material, we usually see the reflected light as composed of two distinct colors that typify the highlight areas and the matte object parts. This is a characteristic property of many dielectrics, and our reflection model capitalizes on this characteristic color change between matte and highlight areas.

When light hits the surface of a dielectric material, the change in refraction indices causes the light to partially reflect back into the air and to partially refract, penetrating into the material body. We refer to the reflection process at the material surface as *surface reflection*. It generally appears as a highlight or as gloss on an object. Fresnel's law describes how the reflected light depends on the refraction indices of the material and the surrounding medium, on the incidence angle and on the polarization of the light[10]. When modelling surface reflection, however, it is common to make some simplifying assumptions. Assuming that little or no light absorption occurs at the material surface, we may conclude that the light reflected from the surface has the same color as the illumination. This is a characteristic feature of highlights on most dielectric materials. Depending on the smoothness of the surface, the light may be reflected in a preferred direction (mirror reflection) or it may be scattered into many directions. Several models have been developed in the physics and computer graphics communities to describe the geometric properties of light reflection from rough surfaces[11, 12, 13].

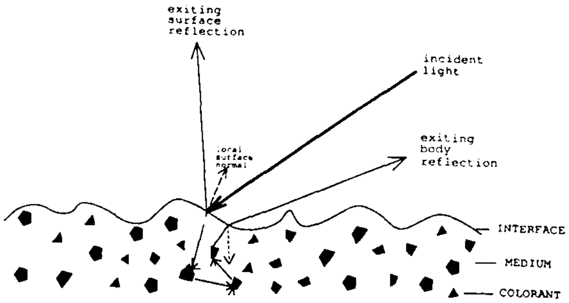


Figure 2-1: Reflection from a dielectric material

For dielectric materials, not all incident light is reflected at the material surface. Some percentage of the light penetrates into the material body. The refracted light beam travels through the medium, hitting pigments from time to time. The pigments scatter the light and partially or entirely absorb it at some wavelengths[14, 15]. Some of the light finally exits from the material body back into the air. We refer to this reflection process as *body reflection*. Its geometric and photometric properties depend on many factors: the transmitting properties of the medium, the scattering and absorption properties of the pigments, and the shape and distribution (including density and orientation) of the pigments. If we assume a random distribution of the pigments, the light exits in random directions from the body. In the extreme, when the exiting light is uniformly distributed, the distribution can be described by Lambert's law. The distribution of the pigments also influences the amount and the color of the reflected light. If the pigments are distributed randomly in the material body, we may expect that, on the average, the same amount and color will be absorbed everywhere in the material before the light exits. In such a case, the light that is reflected from the material body has the same color over the entire surface.

According to the above discussion, our reflection model describes the light, $L(\lambda,i,e,g)$[2], which is reflected from an object point as a mixture of the light $L_s(\lambda,i,e,g)$ reflected at the material surface and the light $L_b(\lambda,i,e,g)$ reflected from the material body.

$$L(\lambda,i,e,g) = L_s(\lambda,i,e,g) + L_b(\lambda,i,e,g) \qquad (1)$$

If we assume that there is only a single light source in the scene, with no inter-reflection between objects, and that highlights (surface reflection) have the same color as the illumination, we can then separate the spectral reflection properties of L_s from its geometric reflection properties. We thus model it as a product of a spectral power distribution, $c_s(\lambda)$, and a geometric scale factor, $m_s(i,e,g)$, which describes the intensity of the reflected light. Similarly, we separate the body reflection component L_b into a spectral power distribution, $c_b(\lambda)$, and a geometric scale factor, $m_b(i,e,g)$. Substituting these terms into equation (1), we obtain the Dichromatic Reflection Model equation:

$$L(\lambda,i,e,g) = m_s(i,e,g)c_s(\lambda) + m_b(i,e,g)c_b(\lambda) \qquad (2)$$

We thus describe the light that is reflected from an object point as a mixture of two distinct spectral power distributions, $c_s(\lambda)$ and $c_b(\lambda)$, each of which is scaled according to the geometric reflection properties of surface and body reflection. In the infinite-dimensional vector space of spectral power distributions (each wavelength defines an independent dimension in this vector space[16, 17]), the reflected light can thus be described as a linear combination of the two vectors $c_s(\lambda)$ and $c_b(\lambda)$.

Many reflection models, which have been developed in the physics and computer graphics communities[18, 4, 12, 13] are special cases of the model described here[6]. They replace the geometric variables, m_s and m_b, by specific functions that approximate the measured reflection data of a chosen set of typical materials. In our work, we concentrate on the spectral variables in equation 2, $c_s(\lambda)$ and $c_b(\lambda)$, exploiting the color difference between them. We leave the geometrical factors unspecified.

3. Object Shape and Color Variation

We will now discuss the relationship between the light mixtures of all points on an object. We study the spectral variation over an entire object by analyzing the *histogram of the light mixtures from all object points*.

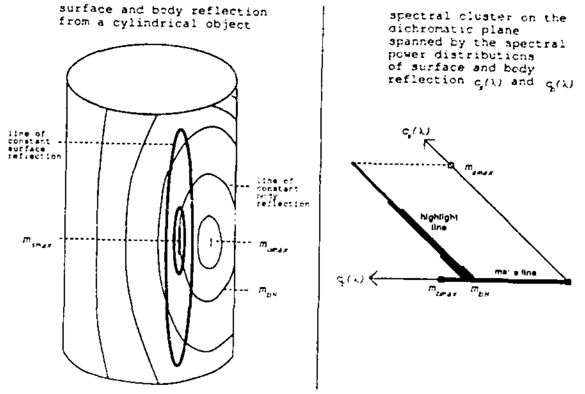

surface and body reflection from a cylindrical object
line of constant surface reflection
line of constant body reflection
m_{smax}
m_{bmax}
m_{bH}
spectral cluster on the dichromatic plane spanned by the spectral power distributions of surface and body reflection $c_s(\lambda)$ and $c_b(\lambda)$
$c_s(\lambda)$
m_{smax}
highlight line
matte line
$c_b(\lambda)$
m_{bmax}
m_{bH}

Figure 3-1: The shape of the spectral cluster for a cylindrical object

An investigation of the geometrical properties of surface and body reflection reveals that the light mixtures form a dense spectral cluster in the dichromatic plane. The shape of this cluster is closely related to the shape of the object, as we will now describe. For illustration purposes, we will assume in the following discussion of spectral histograms that body reflection is approximately Lambertian and that surface reflection is describable by a function with a sharp peak around the angle of perfect mirror reflection. Note, however, that this analysis is not limited to a particular geometric reflection model. Figure 3-1 shows a sketch of a shiny cylinder. The left part of the figure displays the magnitudes of the body and surface

[2] i,e, and g describe the angles of the incident and emitted light and the phase angle; λ is the wavelength parameter.

reflection components. The curves show the loci of constant body or surface reflection. The darker curves are the loci of constant surface reflection. Since $m_s(i,e,g)$ decreases sharply around the object point with *maximal surface reflection*, m_{smax}, these curves are shown only in a small object area. We call the points in this area *highlight points*. The remaining object points are *matte points*. The right part of the figure shows the corresponding spectral histogram in the dichromatic plane. As we will describe below, the object points form two linear clusters in the histogram.

For **matte points**, the surface reflection component of the reflected light is negligible and all the reflected light comes from body reflection. The observed light at such points thus depends only on $c_b(\lambda)$, scaled by $m_b(i,e,g)$ according to the geometrical relationship between the local surface normal of the object and the viewing and illumination directions. Consequently, the matte points form a *matte line* in the dichromatic plane in the direction of the body reflection vector, $c_b(\lambda)$, as shown in the right part of Figure 3-1.

Highlight points exhibit both body reflection and surface reflection. However, since $m_s(i,e,g)$ is much more sensitive to a small change in the photometric angles than $m_b(i,e,g)$, the body reflection component is generally approximately constant in a highlight area, as displayed by the curve with label m_{bH} in Figure 3-1. Accordingly, the second term of the Dichromatic Reflection Model equation (2) has a constant value, $m_{bH}\, c_b(\lambda)$, and all spectral variation within the highlight comes from varying amounts of $m_s(i,e,g)$. The highlight points thus form a *highlight line* in the dichromatic plane in the direction of the surface reflection vector, $c_s(\lambda)$. The line departs from the matte line at position $m_{bH}c_b(\lambda)$, as shown in Figure 3-1. More precisely, the highlight cluster looks like a slim, skewed wedge because of the small variation of the body reflection component over the highlight.

The combined spectral cluster of matte and highlight points thus looks like a skewed T. The skewing angle of the T depends on the spectral difference between the body and surface reflection vectors while the position and width of the highlight line depend on the illumination geometry: If the phase angle g between the illumination and viewing direction at a highlight is very small, the incidence direction of the light is close to the surface normal. The underlying amount of body reflection thus is very high. The highlight line then starts near the tip of the matte line, and the skewed T becomes a skewed L. The wider the phase angle g is, the smaller is the amount of underlying body reflection. We have investigated this relationship more precisely for the case of a spherical object which is illuminated by a point light source at some distance d and viewed by a camera from another point at the same distance from the object. If the angle between the illumination and the viewing direction becomes too wide, the object point with (globally) maximal body reflection becomes invisible to the camera. For wider phase angles, the (local) maximum in body reflection, $m_{b.LocalMax}$, decreases as the phase angle increases. We have calculated the relationship between $m_{b.LocalMax}$ and the amount of body reflection under the highlight, m_{mH}, for varying illumination geometries. Under reasonable imaging conditions, m_{bH} has a value that is more than 48% of $m_{b.LocalMax}$. The highlight line thus generally starts in the upper 50% of the matte line. A more detailed analysis will be presented in[19]. We exploit this property as the *50%-heuristic* in our segmentation method.

4. A Camera Model

The previous sections have described light reflection in a theoretical, physical model. However, the observed pixels values are also influenced by the characteristics of the recording camera. Since some of these influences disturb the light reflection properties stated in the Dichromatic Reflection Model, we need to provide methods that restore the physical properties of light reflection. Where this is impossible, our image analysis algorithms must be able to detect or tolerate the inaccuracies in the image data. This section briefly describes how some camera characteristics influence the pixel values in real images. A more detailed analysis, including color pictures with color clusters from real images, can be found in[7].

The Dichromatic Reflection Model describes light reflection using the continuous spectrum of light. When a sensing device such as a camera or the human eye records an image, the light is integrated over the spectrum using a small number of weighting functions such as color filters. This process of *spectral integration* sums the amount of incoming light, $L(\lambda,i,e,g)$, weighted by the spectral transmittance of the filter, $\tau_f\,(\lambda)$, and the spectral responsivity of the camera, $s(\lambda)$, over all wavelengths λ:

$$C_f = \int L(\lambda,i,e,g)\tau_f(\lambda)s(\lambda)d\lambda \qquad (3)$$

We use a red, a green and a blue color filter (Wratten filters number 25, 58 and 47A), thus reducing the infinite-dimensional vector space to a three-dimensional space. The spectrum of an incoming light beam at pixel position (x,y) is represented by a triple $\mathbf{C}(x,y) = [R,G,B]$, where i, e, and g are determined by x and y and by the position of the object relative to the camera.

Spectral integration is a linear transformation[20, 17]. For this reason, the linear relationship between reflected light and the colors of surface and body reflection, as stated in equation (2), is maintained under spectral integration. We thus obtain the Dichromatic Reflection Model for the three-dimensional color space:

$$\mathbf{C}(x,y) = m_s(i,e,g)\mathbf{C_s} + m_b(i,e,g)\mathbf{C_b} \qquad (4)$$

The color pixel value **C***(x,y)* is thus a linear combination of the two color vectors, $\mathbf{C_s} = [R_s,G_s,B_s]$ and $\mathbf{C_b} = [R_b,G_b,B_b]$, which describe the colors of surface and body reflection on an object in the scene. Within the three-dimensional color space, $\mathbf{C_s}$ and $\mathbf{C_b}$ span a dichromatic plane which contains a parallelogram in which the color pixel values lie.

We will now briefly describe a few camera limitations that are important factors when images of scenes with highlights are taken.

- Real cameras have only a limited dynamic range to sense the brightness of the incoming light. This restricts our analysis of light reflection to a *color cube*, as shown in Figure 4-1. If the incoming light is too bright at some pixel position, the camera cannot sense and represent it adequately and the light is *clipped* in one or more color bands. Color clipping can be a problem for measuring the color of highlights or bright objects. In the color histograms, it causes the clusters to bend along the walls and edges of the color cube (see Figure 4-1). Such *clipped color pixels* do not follow the characteristics of the dichromatic reflection model and must thus be distinguished from matte pixels and highlight pixels.

- If a CCD-camera is used to obtain the images, too much incident light at a pixel may completely saturate the sensor element at this position[21]. This causes *blooming* in the camera[22] as a result of which adjacent pixels increase their values proportional to the magnitude of the overload. We call such neighboring pixels *bloomed color pixels*. We suspect color values in the upper 10% of the intensity scale in our images to be potentially influenced by color clipping or blooming.

- Cameras are generally much less sensitive to blue light than to red light. In order to provide an equal scaling on the three color axes in the color space, we need to rescale the pixel data separately in the color bands. We refer to this procedure as *color balancing*. We balance the color bands by controlling the camera aperture during the picture taking process, using apertures for greer and blue exposures under tungsten light that are *1/2* and *1 1/2* f-stops higher than the aperture used for the red exposure. We also use a total IR suppressor (Corion FR-400) in front of our camera to eliminate the CCD-camera's sensitivity to infrared light.

- The color pixels also depend on the camera response to incident light flux. Due to *gamma-correction*, the camera output is generally related by an inverse power law to the incident flux. It introduces curvature into the color space, distorting the linear properties of the Dichromatic Reflection Model. To linearize the color data, we fit interpolating cubic splines to the measured responsivity data, separately in each color band, as suggested by LeClerc[23]. To model very bright light reflection, as can occur in highlights, we rescale the linearization function and extrapolate it from the brightest measurement outwards by a square root curve. We use these functions to generate a look-up table relating the measured intensities in every color band to the incident flux.

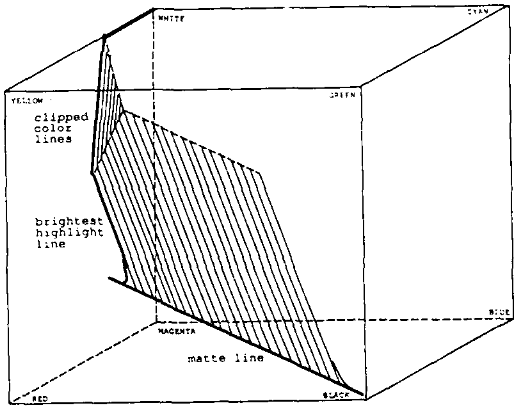


Figure 4-1: Color cluster in the color cube, with color clipping

5. Color Image Segmentation

We will now describe the automatic segmentation method that we have developed based on the Dichromatic Reflection Model and the camera model described above. The goal of segmentation is to identify objects in an image, as delimited by material boundaries. Because most current color segmentation methods are based on a very simple interpretation of color changes in an image, they generally segment images not only along material boundaries but also along other lines exhibiting color or intensity variations, such as highlight and shadow boundaries, or object edges with significant shading changes. The Dichromatic Reflection Model provides a more sophisticated interpretation scheme relating the physics of light reflection to color changes in the image and in the color space. Our segmentation algorithm uses the Dichromatic Reflection Model and is thus able to distinguish significant color changes at material boundaries from insignificant ones that are due to shading changes or highlights. The algorithm generates hypotheses about skewed T's in the color cube while it analyzes local color variations in the image. It then uses the hypothesized orientations of the color clusters to decide whether a color change is consistent with the local skewed-T model or whether it constitutes a material change.

5.1. Generating Initial Hypotheses about Color Clusters

We may use either a global (top-down) or a local (bottom-up) approach to determine skewed T's in the color histogram and the associated regions in the image. A global algorithm projects the entire image into the color space and then applies some analysis technique to the color space to identify and distinguish between the various clusters. A local algorithm, on the other hand, starts out with small image areas and merges areas with consistent interpretations into larger areas, assuming that local color variation is related to only a few scene properties. The global approach has problems when several objects with very similar colors exist in the scene; a local approach, on the other hand, has the problem of distinguishing camera noise from systema'ic color variation.

Figure 5-1 shows a scene with eight plastic objects under white light. The scene contains a cup and two donuts at various shades of red, a cup and a donut in different yellow colors, a green cup and a green donut, and a blue donut. The upper left quarter displays the original image. The color histogram from the entire image is shown in the upper right quarter. Note that the various clusters overlap significantly. In such an image, we feel that it is harder to decide when to split a large region with overlapping color clusters than to accomodate for camera noise. Therefore, we chose to utilize a local (region growing) scheme.

We start by dividing the image into small, non-overlapping windows of a given size (typically 10x10 pixels). We project the color pixels from one window at a time into the color space and find the principal components of the color distributions, as indicated by the eigenvectors and eigenvalues of the covariance matrix of the cluster[24, 25]. We sort the eigenvalues by decreasing size. The eigenvalues and eigenvectors determine the orientation and extent of the ellipsoid that optimally fits the data.

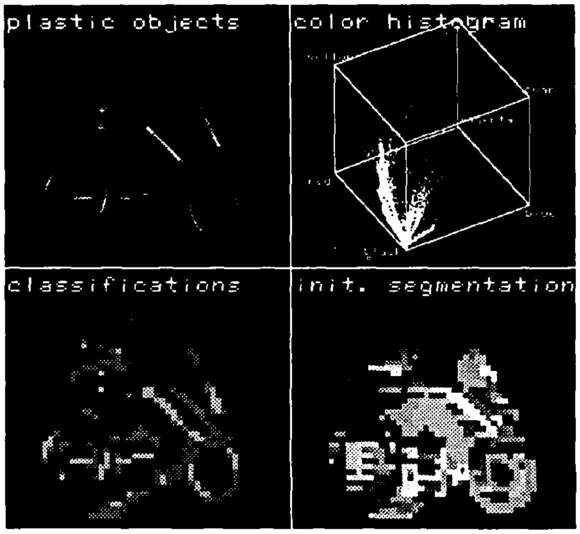


Figure 5-1: Initial analysis of color variations for a scene with eight plastic objects

The shape of the ellipsoids provides information that we can use to relate the local color variation to physical interpretations. The classification is based on the number of eigenvalues that are approximately zero, within the limit of pixel value noise in the image. In order to classify the ellipsoids, we compare their eigenvalues with the estimated amount of noise, σ_0. For our camera, we use an experimentally determined estimate of $\sigma_0 = 2.5$. We base our decision for each eigenvalue on a χ^2-test with (n^2-1) parameters, where n is the window size, and we use a confidence level of $\alpha_0 = 0.005$. We then classify each cluster according to how many eigenvalues are determined to be significantly greater than zero.

- In zero-dimensional (*pointlike*) clusters, all three eigenvalues of the window are very small and the window probably lies on a single object which is either very flat or very dark.

- One-dimensional (*linear*) clusters are clusters for which only the first eigenvalue is significantly larger than the estimated camera noise. Pixels in such a window may stem from a matte object area or from the interior of a highlight area such that they form part of a matte or highlight cluster. The pixels may also come from two point clusters if a window overlays parts of two neighboring objects.
- Two-dimensional (*planar*) clusters have large first and second eigenvalues, and the local color data thus fits a plane in the color cube. Such clusters occur at windows that either cover some matte and some highlight pixels of one object or that overlay matte pixels from two neighboring regions.
- In three-dimensional (*volumetric*) clusters, all three eigenvalues are large. Such color clusters may arise in the middle of highlights where color clipping and blooming significantly increase the noise in the pixel measurements. Volumetric color clusters also occur along material boundaries when three or more objects in different colors share a window or when a window overlays matte pixels of one object and matte and highlight pixels of another object.

The lower left quarter in Figure 5-1 shows the classification of the color clusters from the initial 10x10 windows. Pointlike clusters are displayed in black, linear clusters in dark grey, planar clusters in light grey, and volumetric clusters in white. The image shows that the classifications relate in the expected way to scene properties. Most matte object areas are covered by linear windows, while windows at material boundaries and at highlights are planar or volumetric.

Next, we merge neighboring windows that have similar color characteristics. The algorithm proceeds in row-major order. For every area, it tests for every neighbor whether the two can be merged. In order not to merge windows across material boundaries, we only merge neighboring windows, if both of them and the resulting larger window all have the same classification. We use the χ^2-test presented above to classify the larger window. Accordingly, we combine windows with pointlike clusters into larger pointlike windows; we merge windows with linear characteristics into larger linear windows; and we merge planar windows into larger planar windows. We do not merge neighboring volumetric color clusters since there is no constraint on the resulting cluster. We continue this process until no more areas can be merged. The results are initial hypotheses about the positions and orientations of pointlike, linear, and planar clusters in color space and their respective approximate extents in the image. The lower right quarter in Figure 5-1 presents the results of merging neighboring windows of the same class in the image of the eight plastic objects.

5.2. Exploiting Linear Hypotheses

So far, the algorithm has used a bottom-up approach to extract information about the scene from the image. In a top-down step, the algorithm now uses the generated hypotheses to segment the image. We start by using the hypothesis with the largest linear cluster. The chosen linear hypothesis provides us with a model of what color variation to expect in the associated image area. The mean value and the first eigenvector describe the position and orientation of a linear color cluster, while the second and third eigenvalues determine the extent of the color cluster perpendicular to the major direction of variation. According to the Dichromatic Reflection Model, we attribute color variation along the major axis to a physical property of the scene, such as a changing amount of body or surface reflection or a material boundary.

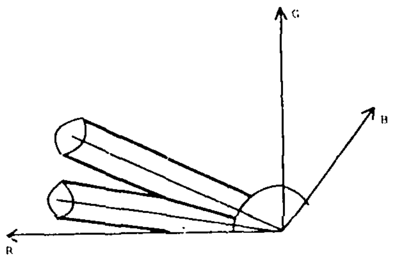


Figure 5-2: Resolving a color conflict between two matte color clusters

Our model assumes that color variation perpendicular to the first eigenvector is caused by random noise. We thus approximate a linear color cluster by a cylinder. The radius depends on the estimated camera noise; we generally choose a constant multiple of σ_0 (typically $4\sigma_0$). We exclude dark color pixels from our color analysis because the matte clusters merge near the dark corner of the color cube. The cylinder is thus delimited at its dark end by a sphere which is centered at the black corner. We typically use a radius for the sphere of 23, which is approximately 10% of the maximum pixel value.

We then use the linear hypothesis to locally resegment the image. We select a start pixel from the area associated with the color cluster: this pixel must have a color that is contained within the color cylinder of the current hypothesis. We then examine its four neighbors, testing whether their colors lie on the cylinder. If so, we recursely examine the next fringe of neighbors and so on. We thus grow a region of four-connected image pixels that are consistent with the current hypothesis.

Since the matte clusters converge towards the dark corner of the color cube, there exists a potential conflict between neighboring matte areas. Our heuristic of excluding very dark pixels from the analysis eliminates the most difficult cases; still, the cylinders of neighboring clusters may intersect beyond the selected dark threshold. This depends on the cylinder radius and on the angle between the two cylinders. Thus, neighboring objects with similar colors will have conflicts even at fairly bright colors. We assign pixels with a color conflict to the cluster with the closest axis, as shown in Figure 5-2.

Since there is generally more than one object in the scene, our algorithm iterates the above steps for each image area with a large linear cluster, selecting the areas by decreasing size. It stops when the next selected area is too small (typically less than 500 pixels).

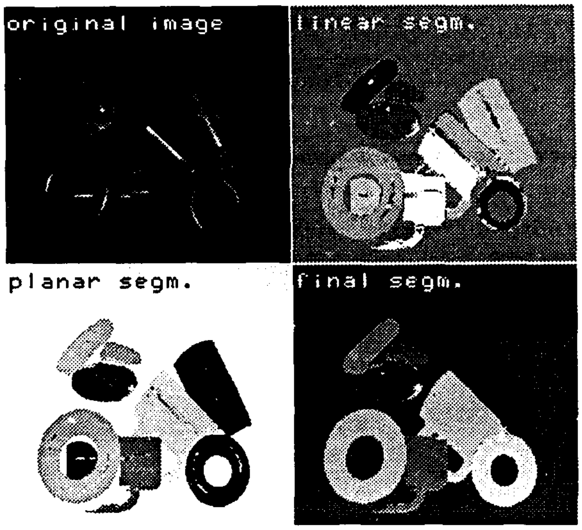


Figure 5-3: Segmentation steps of the scene with eight plastic objects

In principle, the resulting "linear" regions may be related in any of several different ways to the physical processes in the scene. As discussed in section 5.1., a linear color cluster may be a matte cluster or a highlight cluster or even a combination of two clusters across a material boundary. However, we expect that linear color clusters from highlights and across material boundaries are generally much smaller than matte clusters. Since we use only hypotheses connected with large linear clusters, we assume in the following that all of the linear regions correspond to matte linear clusters.

The upper right quarter of Figure 5-3 shows the results of selecting and applying linear hypotheses to the image with the eight plastic objects. The region boundaries outline the matte object parts in the scene, with the material boundaries being well observed. The highlight areas on the objects have not yet been analyzed. This sometimes divides the matte pixels on an object into several matte areas, as shown on the middle cup and on the lower right donut.

5.3. Generating Planar Hypotheses

We will now extend the above generated linear hypotheses into planar hypotheses that describe dichromatic planes and skewed T's. For every linear region, we examine all its neighbors to determine a prospective highlight candidate. In order to decide whether a neighboring region comes from another object (across a material boundary) or whether it is a highlight on the same object, we test whether both linear clusters point in positive directions and whether the two clusters form a skewed T and meet in the upper 50% of the matte cluster. A similar idea was used by Gershon[8] to identify highlight segments in a postprocessing step after an initial, traditional segmentation was performed. He tested whether the two clusters are nearly parallel or whether they intersect, suspecting that parallel clusters are neighboring matte clusters. In our method, the physical model is used during the segmentation process rather than as a post-processing step.

Each candidate cluster that we test is a group of pixels on the border of a linear region. If the cluster is in fact a highlight, the first eigenvector of the cluster's color distribution will generally be in the direction of the highlight color. We start by testing whether the first eigenvector of a highlight candidate describes a positive color change, i.e: $dR \geq 0$ and $dG \geq 0$ and $dB \geq 0$. If this is not the case, the window of the highlight candidate probably overlays a material boundary and thus partially covers pixels from the current matte region and partially covers pixels from a neighboring matte region.

Next, we exploit the T-shape of the dichromatic cluster and the 50%-heuristic. For this purpose, we determine the brightest matte point in the color cluster. In order not to select a highlight point that fell into the matte cylinder, as shown in Figure 5-4, we exploit the observation that highlight clusters always grow inwards into the color cube, due to the additive nature of body and surface reflection. We thus choose the brightest matte point only from pixels with color values that are closer to the walls of the cube than the matte line is. Next, we determine the intersection points of the two clusters. We use their color means and first eigenvectors to describe a matte and a prospective highlight line and determines the two points on the two lines that are closest to one another. If the distance between them is larger than a multiple of the estimated camera noise (typically $4\sigma_0$), we decide that the clusters do not meet in a skewed T and the neighboring area is discarded. Similarly, if the clusters intersect in the lower 50% of the matte cluster, we suspect that we are looking at two matte clusters from neighboring objects, and we discard this neighbor.

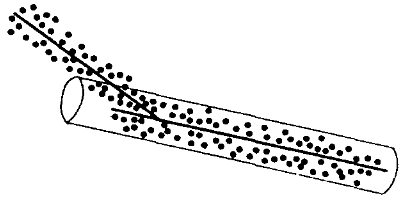


Figure 5-4: Finding the brightest matte pixel

There may be several highlight candidates because the highlight on the object may consist of a series of small windows that could not be merged, due to color clipping and blooming in the middle of the highlight. In order to select a good representative of the entire highlight, we average the intersection points of all highlight candidates, weighted by the number of pixels in the various regions. We then select the highlight region whose intersection point is closest to the average.

Since we assume in the Dichromatic Reflection Model that highlights have the same color as the illumination, all highlight clusters are parallel to one another and to the illumination color vector. As a consequence, all dichromatic planes intersect along one line, which is the illumination vector. We use this constraint to further reduce the error estimating the orientations of the highlight clusters, combining all highlight hypotheses intc a single hypothesis about the color of the illumination. Observing that any imprecision of the highlight vector orientation within the dichromatic plane is irrelevant we base our computation on the normals $\mathbf{n}_i$ of the dichromatic planes of all objects. We then determine the intersection vector $\mathbf{i}_{i,j}$ between each pair of planes:

$$\mathbf{n}_i = \mathbf{C}_{bi} \times \mathbf{C}_{si}$$
$$\mathbf{i}_{i,j} = \mathbf{n}_i \times \mathbf{n}_j \qquad (5)$$

Each intersection vector $\mathbf{i}_{i,j}$ provides us with a vector describing a hypothesized illumination color. The strength of the hypothesis depends on the angle between the two dichromatic planes under consideration. If the planes intersect nearly perpendicularly, the intersection line will not vary much with small amounts of error in estimating the orientations of the planes. If, on the other hand, the planes are nearly parallel, small amounts of noise may arbitrarily influence whether and where the planes intersect. We use a weighted average of all intersection vectors to generate a hypothesis on the illumination color, in which every intersection vector contributes according to the dihedral angle enclosed by the respective dichromatic planes.

5.4. Exploiting Planar Hypotheses

The Dichromatic Reflection Model states that color variations on a single object lie in a plane in the color space. We therefore use the planar hypotheses to resegment the image, expecting that the resulting image areas cover entire objects and will not be interrupted by highlight boundaries. We select one hypothesis at a time and apply it to the image, proceeding iteratively until no more unprocessed hypotheses with sufficient support from the image exist.

The chosen matte cluster and the illumination hypothesis determine a planar hypothesis predicting all significant color variation on the associated object. We use the cross product of the illumination vector and the first eigenvector of the matte cluster to determine the normal to this dichromatic plane. We use the color mean of the matte cluster to position the plane in the color cube, and we extend the plane into a slice of fixed thickness to account for camera noise. A typical choice for the width of the slice is $4\sigma_0$ in the positive and negative direction of the normal.

The algorithm uses the chosen planar hypotheses to locally resegment the image. In principle, we start from the selected matte region and expand it, recursively including pixels at the region boundaries which are consistent with the planar hypothesis. However, the algorithm must be augmented with special provisions to handle coplanar color clusters from neighboring objects. If the illumination vector lies in the plane spanned by the matte vectors of two neighboring objects, the color clusters of the objects lie in the same dichromatic plane and thus cannot be distinguished by a mere planar region growing approach. The resulting segmentation would generally be quite counterintuitive since the matte object colors may be very different and even complementary. To avoid such segmentation errors, we exploit the previously gathered knowledge about existing matte color clusters. When the planar region process encounters pixels from a previously grown matte region other than the starting region, it only continues growing if the pixel lies within the matte color cylinder of the starting region. We thus apply the unrestricted planar growing criterion only to pixels that have not been previously recognized as matte pixels, while we fall back to the linear region growing method when matte pixels are concerned. This reflects the observation that if several matte areas exist in an object area, separated by highlight regions, all such matte areas form a single matte cluster. We also apply the proximity heuristic described in the previous section to resolve ambiguities for color pixels at the intersection of dichromatic planes.

The lower left quarter of Figure 5-3 displays the results of segmenting the scene using the generated planar hypotheses. In comparison to the linear segmentation in the upper right quarter, the segmented image areas have grown into the highlight areas. As a result, the two matte image areas on the middle cup that were previously separated by a highlight are now united. The same occurred on the lower right donut. Due to camera limitations, not all pixels in the centers of the highlights are yet integrated into the object areas. This will be discussed and remedied in the next section.

5.5. Accounting for Camera Limitations

Unfortunately, real images generally do not comply with the Dichromatic Reflection Model. Color clipping and blooming may distort the color of pixels in and around highlights significantly. As a result, the color pixels in the centers of highlights will generally not fall into the planar slice defined for the dichromatic plane of an object area and the planar segmentation will exclude such pixels, as can be observed in the upper right quarter of Figure 5-3.

Since color information is so unreliable for these pixels, we do not use it. Instead, we use a geometric heuristic to include them into the region. As mentioned above, pixels with distorted colors generally occur in the middle of the highlight areas. Thus, starting from highlight pixels we expand the planar regions into areas that are next to highlight pixels and contain very bright pixels (i.e: brighter than the intersection point between the matte and highlight cluster).

The lower right quarter of Figure 5-3 displays the results of segmenting the scene using the generated planar hypotheses. Nearly all pixels in the highlight centers have now been integrated into the segmented regions, and the image segments correspond very well to the objects in the scene. Very few pixels on the highlights have been excluded, due to our heuristic of only intregrating very bright pixels. Any image processing method for filling holes in image segments should be able to include these pixels. The lowest of the three donuts in the upper left exhibits two interesting properties. First, a small area at its upper left edge has wrongly been assigned to the donut above it. We can see in the original image that this area is covered by a shadow, resulting in very dark pixel values. Since the color clusters of the two donuts (yellow and red) are already merged at these color values, the assignment was based on the distance from the two cylinder axes, which happened to be smaller for the upper donut. Second, there is a small area in the lower right part of the donut that was not included into the large image segment covering the donut. The color of these pixels has been significantly altered by inter-reflection from the middle cup which reflected a part of its body color (orange) onto the (yellow) donut, thus influencing the reflected colors of the donut. Inter-reflection may also be a factor in the shadowy part in the upper left of the donut. This demonstrates the sensitivity of our algorithm to the reflection processes in the scene. Since inter-reflection is not yet a part of our reflection model, our current algorithm cannot identify it in the image and process it correctly.

6. Removing Highlights

As one application of our above method to analyze and segment color images, we can now use the gathered information about the color clusters to split every color pixel into its two reflection components. We thus generate two intrinsic images of the scene, one showing the objects as if they were completely matte, and the other showing only the highlights.

We have previously reported a method to detect and remove highlights from hand-segmented images [7]. That method projected the pixels from a selected image area into the color space and fitted a skewed T to the entire color cluster, thus determining the body and surface reflection vectors, $\mathbf{C_b}$ and $\mathbf{C_s}$, of the area. Since our new segmentation algorithm already provides this information as a result of its analysis of local color variations, we can now skip this step. However, due to possible estimation errors in the segmentation process and due to camera problems such as blooming and chromatic aberration, the vectors may not yet fit perfectly. In order to obtain a more precise fit to the data, we retest every pixel in the segmented area and label it as a matte or highlight pixel, depending on whether it is closer to the matte line or to the highlight line, which is the illumination vector starting from the intersection point of the skewed T. We then refit the matte and highlight line to the matte and highlight pixels by determining the first eigenvectors and the color means of the clusters.

The algorithm uses the reflection vectors $\mathbf{C_b}$ and $\mathbf{C_s}$ that are defined for every segmented region to split every pixel in each region into its two reflection components. $\mathbf{C_b}$, $\mathbf{C_s}$, and their cross product, $\mathbf{C_b} \times \mathbf{C_s}$, define a new (not necessarily orthogonal) coordinate system in the color cube. This coordinate system describes every color in the cube in terms of the amounts of body reflection, surface reflection and noise ε, as given by the color distance from the dichromatic plane (see figure 6-1). There exists an affine transformation, and thus a linear transformation matrix *T*, which transforms any color vector **c** = $[R,G,B^T]$ from the initial coordinate system into a vector **d** = $[m_b,m_s,\varepsilon^T]$ in the new coordinate system. After computing *T* from $\mathbf{C_b}$ and $\mathbf{C_s}$, we can thus transform every color pixel in the image area into its constituent body and surface reflection components, m_b and m_s. By selecting the m_b-component of every transformed pixel, we generate the *body reflection image* of the image area. By selecting the m_s-component, we generate the corresponding *surface reflection image.*

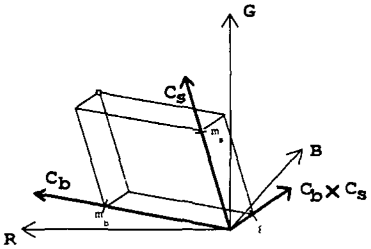


Figure 6-1: Decomposing a color pixel into its constituent reflection components

We cannot apply this method to bloomed and clipped pixels because such pixels generally do not satisfy the conditions of the Dichromatic Reflection Model: since their color is altered by the sensing process in the camera, they generally do not lie on the skewed T or even in the dichromatic plane. In order to restore the physically correct color of such pixels, we exploit the observation that in many cases, clipping and blooming occurs only in one or two color bands. The pixels may thus have correct data in the other color bands. We assume that the smallest of the three values of a color pixel stems from a color band without clipping and blooming. We then replace the clipped or bloomed pixel by a pixel on the matte or highlight line that has the same value in the undistorted band.

The upper quarters in Figure 6-2 display the resulting intrinsic images of the scene with the eight plastic objects. The images demonstrate that we are able to determine the body and surface reflection components of the various objects reasonably well. The orientations of the corresponding reflection vectors are shown in Table 9-1. When evaluated under eyeball inspection, the body reflection image seems to generally provide a smooth shading component across the highlight area. We expect that this image may therefore be a useful tool to determine object shapes from shading information. We will investigate this application in the future. There exist thin dark rings in the body reflection image around the highlight on the lower right donut. This error is due to chromatic aberration in the camera which currently limits the performance of our algorithm.

The surface reflection image exhibits very well the highlight components on the objects. All highlights have been detected, and beyond that, the image provides precise quantitative information on the varying amount of surface reflection. Notice the gradient in the surface reflection component on the rightmost cup and the small amount of gloss that was detected on the handle of the middle cup. A careful inspection of the surface reflection image reveals that the surface reflection component also increases at the material boundaries. This effect is related to aliasing occuring at pixels that integrate light from two neighboring objects. The colors of such pixels are a linear combination of two matte object colors. Depending on the orientations of the dichromatic planes, they may be included in either of the two object areas in the image, or they may be left unassigned, as at the edge between the middle and the rightmost cup. If they are assigned to one of the object areas, they generally do not lie close to the matte line, thus resulting in a higher surface reflection component.

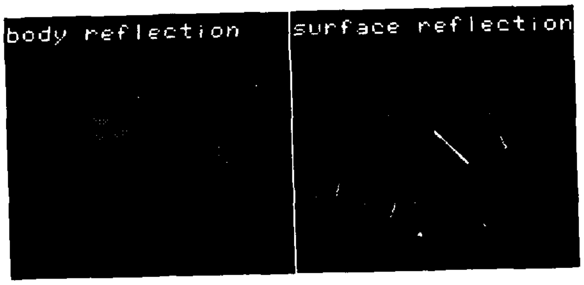


Figure 6-2: Intrinsic reflection images of the scene with eight plastic objects

7. Results and Discussion

Figure 7-1 shows the results of applying our segmentation algorithm and the reflection analysis to a scene of an orange, a green and a yellow cup under white light. These same images were analyzed in our earlier work, using hand-segmentation[7]. The segmented image in the upper right quarter demonstrates that our method is capable of correctly identifying the object areas. Due to its modeling of matte and highlight color variations, our program ignores color changes along the highlight boundaries. The lower quarters in the figure display the intrinsic body and surface reflection images. The orientations of the matte and highlight vectors are given in Table 9-2. Table 9-3 shows the vector orientations of the cups under yellow light. The intrinsic images and the reflection vectors are comparable or better than the ones that we obtained earlier by fitting skewed Ts to the entire histogram of hand-segmented image areas [7].

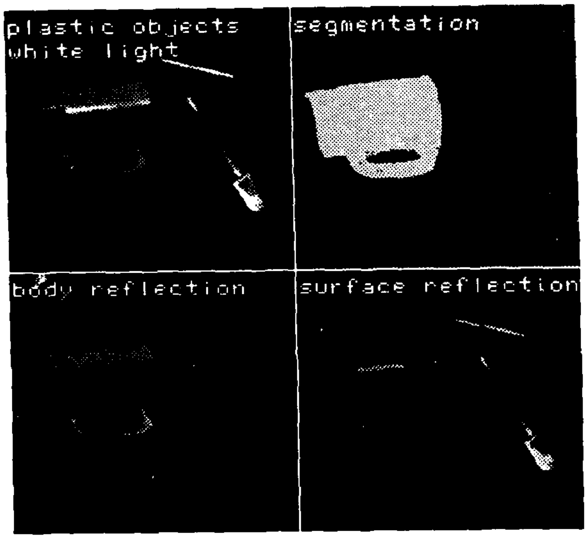


Figure 7-1: Analysis results for three plastic cups under white light

Our examples show that our algorithm analyzes and segments the given images quite well. We will now discuss the conceptual and implementational limitations of our approach. First, our current implementation depends on the window size used for determining the initial local color variation in the image. If the windows are too large, many of them overlap several objects, and the clusters are volumetric. If the windows are too small, color variation on a relatively flat or dark object may be overlooked, as the result of pointlike clusters. Second, our cylindrical model for the linear hypotheses does not account for secondary effects on pixel values, such as color clipping, blooming, chromatic aberration, aliasing and inter-reflection between objects. Because such effects influence the shape of the color cluster, a non-circular model of the cross-section of linear clusters may be needed. We deal with this problem by choosing a fairly large cylinder radius. It may be desirable to adaptively fit a tighter model to the linear cluster. Along the same lines, the hypothesized orientation of the linear cluster may be inexact and could be improved by an adaptive method that refits the cylinder until it optimally fits a given cluster. We are currently investigating these issues.

The conceptional limitations of our approach are related to the basic principles of our model. Since we attribute any linear color variation to the changing illumination geometry of a single material, we are unable to find material boundaries between objects with collinear matte clusters. We will need a geometrical analysis, linking intensity gradients to object shape, to distinguish between such objects. The same will be needed to analyze dark image areas which are currently exluded because their color information is too unreliable.

The model also makes simplifying assumptions about the illumination conditions and the materials in the scene. A color cluster from a single object in an unconstrained scene will generally not be a skewed T composed of linear subclusters because the illumination color may vary on different parts on the object surface, and the reflection properties of the object may also change, due to pigment variations in the material body. The necessary extensions to the model will be the subject of future work.

Furthermore, our method to split color pixels into their reflection components (but not the segmentation) relies on a characteristic color change between the matte object color and the highlight color. There needs to be a certain angle between the orientations of the body and surface reflection vectors of an object. How big the angle needs to be depends on the estimated camera noise (as related to the width of the matte cylinder). If the matte and highlight clusters are approximately collinear, we cannot separate the reflection components. Similarly, we have problems when one of the two linear clusters does not exist or is very small for an object. The matte cluster is missing, if the viewed object is very dark, or if the scene is illuminated with a narrow-band illuminant that does not overlap with the wavelengths at which the material reflects light. Matte clusters also do not exist for metallic objects. On the other hand, the highlight cluster may be missing, if an object does not reflect a highlight into the camera, due to its position in the scene and the illumination geometry. As a third case, we need to consider objects with very rough surfaces such that every pixel in the image area has both a significant body and surface reflection component. The color cluster may then fill out the entire dichromatic plane. A common special case of this are so-called "matte" or "lambertian" materials - as opposed to glossy materials - which reflect a constant amount of surface reflection in every direction and thus never exhibit a highlight in the common sense of the word. The corresponding color clusters are linear clusters, translated from the origin of the color space according to the constant surface reflection component. Our current method is not capable of distinguishing between all these cases that result in a single, linear cluster. In combination with exploiting previously determined scene properties, such as the illumination color, we will need to analyze the intensity gradients along the linear axes and relate them to the properties of m_s and m_b, as described in a geometrical model of light reflection.

8. Conclusions

In this paper, we have demonstrated that it is possible to analyze and segment real color images by using a physics-based color reflection model. Our model accounts for highlight reflection and matte shading, as well as for some characteristics of cameras. By developing a physical description of color variation in color images, we have developed a method to automatically segment an image while generating hypotheses about the scene. We then use the knowledge we have gained to separate highlight reflection from matte object reflection. The resulting intrinsic reflection images have a simpler relationship to the illumination geometry than the original image and may thus improve the results of many other computer vision algorithms, such as motion analysis, stereo vision, and shape from shading or highlights [26, 9, 4, 27]. Since the surface reflection component of dielectric materials generally has the same color as the illumination, we can also determine the illumination color from the intrinsic surface reflection image, information which is needed by color constancy algorithms[28, 29, 30, 31].

The key points leading to the success of this work are our modeling of highlights as a linear combination of both body and surface reflection and our modeling of the camera properties. With few exceptions [28, 8, 30, 5], previous work on image segmentation and highlight detection has assumed that the color of highlight pixels is completely unrelated to the object color. This assumption would result in two unconnected clusters in the color space: one line or ellipsoid representing the object color and one point or sphere representing the highlight color. Our model and our color histograms demonstrate that, in real scenes, a transition area exists on the objects from purely matte areas to the spot that is generally considered to be the highlight. This transition area determines the characteristic shapes of the color clusters which is the information that we use to distinguish highlight boundaries from material boundaries and to detect and remove highlights. This view of highlights should open the way for quantitative shape-from-gloss analysis, as opposed to binary methods based on thresholding intensity.

By modeling the camera properties, we are able to obtain high quality color images (through color balancing and spectral linearization) in which most pixels maintain the linear properties of light reflection, as described in the Dichromatic Reflection Model. We can also detect most distorted color pixels in an image and thus generate an intrinsic error image which then guides our algorithm to separate only undistorted color pixels into their reflection components. We expect that the intrinsic error image will be similarly useful in guiding other computer vision algorithms, such as shape from shading. It may also enable us to automatically control the camera aperture so that we can obtain color images with minimal clipping and blooming.

Our hypothesis-based approach towards image segmentation may provide a new paradigm for low-level image understanding. Our method gains its strength from using an intrinsic model of physical processes that occur in the scene. The result are intrinsic images and hypotheses which are closely related in their interpretation to the intrinsic model, being instantiations of concepts formulated in the model. Our system alternates between a bottom-up step which generates hypotheses and a top-down step which applies the hypotheses to the images. Our analysis thus consists of many small, complete interpretation cycles that combine bottom-up processing with feed-back in top-down processing. This approach stands in contrast to traditional image segmentation methods which do not relate their analysis to intrinsic models and that also generally have a strictly bottom-up control structure. We feel that many low-level image understanding methods such as shape-from-x methods, stereo and motion analysis may be viewed and approached under this paradigm. We hope to extend our approach into a more complete low-level image analysis system which combines color analysis with a geometrical analysis of the scene, exploiting the body and surface reflection images. Along these lines, we may generate hypotheses about object shapes and about the object materials[29]. The highlight image may also provide strong evidence for the position of the light source.

Although the current method has only been applied in a laboratory setting, its success shows the value of modeling the physical nature of the visual environment. Our work and the work of others in this area may lead to methods that will free computer vision from its current dependence on statistical signal-based methods for image segmentation.

9. References

1. J.R. Kender, "Instabilities in Color Transformations", *Pattern Recognition and Image Processing,*Vol. PRIP-77, June 1977, pp. 266-274, IEEE Computer Society, Troy NY

2. R. Ohlander, K. Price, and D.R. Reddy, "Picture Segmentation Using a Recursive Region Splitting Method", *Computer Graphics and Image Processing*,Vol. 8, 1978, pp. 313-333.

3. Y. Ohta, T. Kanade, and T. Sakai, "Color Information for Region Segmentation", *Computer Graphics and Image Processing*,Vol. 13, 1980, pp. 222-231.

4. B.K.P. Horn, "Understanding Image Intensities", *Artificial Intelligence*,Vol. 8, No. 11, 1977, pp. 201-231.

5. J.M. Rubin and W.A. Richards, "Color Vision and Image Intensities: When are Changes Material ?", *Biological Cybernetics*,Vol. 45, 1982, pp. 215-226.

6. S.A. Shafer, "Using Color to Separate Reflection Components", *COLOR research and application*,Vol. 10, No. 4, Winter 1985, pp. 210-218, Also available as technical report TR-136, Computer Science Department, University of Rochester, NY, April 1984

7. G.J. Klinker, S.A. Shafer, and T. Kanade, "The Measurement of Highlights in Color Images", *to appear in the International Journal on Computer Vision (IJCV)*, 1988.

8. R. Gershon, *The Use of Color in Computational Vision*, PhD dissertation, Department of Computer Science, University of Toronto, 1987.

9. G. Healey and T.O. Binford, "Local Shape from Specularity". *Proceedings of the First International Conference on Computer Vision (ICCV)*, IEEE, London, June 1987, pp. 151-161.

10. P. Moon, "A Table of Fresnel Reflection", *J. Math. Phys.*,Vol. 19, No. 1, 1940.

11. P. Beckmann and A. Spizzichino, *The Scattering of Electromagnetic Waves from Rough Surfaces*, MacMillan, New York, Vol. 15, 1963, Pages 1-33, 70-96

12. B.T. Phong, "Illumination for Computer Generated Pictures", *Communications of the ACM*,Vol. 18, 1975, pp. 311-317.

13. K.E. Torrance, and E.M. Sparrow, "Theory for Off-Specular Reflection from Roughened Surfaces", *Journal of the Optical Society of America*,Vol. 57, September 1967, pp. 1105-1114.

14. R.S. Hunter, *The Measurement of Appearance*, John Wiley and Sons, New York, 1975.

15. D.B. Judd and G. Wyszecki, *Color in Business, Science and Industry*, John Wiley and Sons, New York, 1975.

16. B.K.P. Horn, "Exact Reproduction of Colored Images", *Computer Vision, Graphics, and Image Processing (CVGIP)*,Vol. 26, 1984, pp. 135-167.

17. S.A. Shafer, "Describing Light Mixtures Through Linear Algebra", *Journal of the Optical Society of America (JOSA)*,Vol. 72, No. 2, February 1982, pp. 299-300.

18. R.L. Cook and K.E. Torrance, "A Reflectance Model for Computer Graphics", *ACM Transactions on Graphics*,Vol. 1, No. 1, January 1982, pp. 7-24, Also published in Computer Graphics 15(3), SIGGRAPH'81

19. G.J. Klinker, *A Physical Approach to Color Image Understanding*, PhD dissertation, Computer Science Department, Carnegie-Mellon University, Summer 1988, In preparation

20. H. Grassmann, "On the Theory of Compound Colors", *Phil. Mag.*,April 1854.

21. W. Budde, *Physical Detectors of Optical Radiation*, Academic Press, New York, Optical Radiation Measurements, Vol. 4, 1983.

22. J.D.E. Beynon and D.R. Lamb (eds.), *Charge-coupled devices and their application*, McGraw-Hill, London, 1980.

23. Y. LeClerc, "A Method for Spectral Linearization", Private communication

24. D.H. Ballard, and C.M. Brown, *Computer Vision*, Prentice Hall, Inc., Englewood Cliffs, NJ 07632, 1982.

25. H.H. Harman, *Modern Factor Analysis, second edition*, University of Chicago Press, Chicago and London, 1967.

26. L. Dreschler and H.-H. Nagel, "Volumetric Model and 3D Trajectory of a Moving Car Derived from Monocular TV Frame Sequences of a Street Scene", *Computer Graphics and Image Processing*,Vol. 20, 1982, pp. 199-228.

27. C.E. Thorpe, *FIDO: Vision and Navigation for a Robot Rover*, PhD dissertation, Computer Science Department, Carnegie-Mellon University, December 1984, Available as technical report CMU-CS-84-168

28. M. D'Zmura and P. Lennie, "Mechanisms of color constancy", *Journal of the Optical Society of America A (JOSA-A)*,Vol. 3, No. 10, October 1986, pp. 1662-1672.

29. G. Healey and T.O. Binford, "The Role and Use of Color in a General Vision System", *DARPA-Image Understanding (IUS) workshop*, L.S. Bauman,ed., Los Angeles, CA, February 1987, pp. 599-613.

30. H.-C. Lee, "Method for computing the scene-illuminant chromaticity from specular highlights", *Journal of the Optical Society of America A (JOSA-A)*,Vol. 3, No. 10, October 1986, pp. 1694-1699.

31. L.T. Maloney and B.A. Wandell, "Color constancy: a method for recovering surface spectral reflectance", *Journal of the Optical Society of America A (JOSA-A)*,Vol. 3, No. 1, January 1986, pp. 29-33.

Reflection Vectors: Eight Plastic Objects under White Light		
	body reflection vector	surface reflection vector
dark red donut	(0.99,0.11,0.12)	(0.62,0.52,0.59)
yellow cup	(0.84,0.52,0.14)	(0.48,0.62,0.62)
green cup (right half)	(0.27,0.89,0.37)	(0.54,0.58,0.61)
red cup	(0.95,0.26,0.14)	(0.68,0.48,0.56)
yellow donut	(0.77,0.61,0.19)	(0.69,0.51,0.51)
bright red donut	(0.98,0.18,0.12)	(0.77,0.61,0.18)
green donut	(0.21,0.78,0.59)	(0.47,0.63,0.31)
green cup (left half)	(0.27,0.86,0.42)	(0.61,0.52,0.59)
blue donut	(0.00,0.25,0.97)	(0.49,0.63,0.60)
illumination vector:		
- computed by alg.		(0.57,0.58,0.57)
- independent meas.		(0.58,0.57,0.58)

Table 9-1: Body and surface reflection vectors of the eight plastic objects under white light

Reflection Vectors: Plastic Cups under White Light		
	body reflection vector	surface reflection vector
green cup	(0.22,0.91,0.37)	(0.46,0.51,0.72)
yellow cup	(0.81,0.57,0.15)	(0.56,0.56,0.61)
orange cup	(0.95,0.26,0.15)	(0.47,0.58,0.66)
illumination vector:		
- computed by alg.		(0.69,0.62,0.38)
- independent meas.		(0.58,0.57,0.58)

Table 9-2: Body and surface reflection vectors of the three plastic cups under white light

Reflection Vectors: Plastic Cups under Yellow Light		
	body reflection vector	surface reflection vector
green cup	(0.49,0.86,0.16)	(0.76,0.62,0.20)
yellow cup	(0.85,0.52,0.08)	(0.74,0.65,0.14)
orange cup	(0.97,0.24,0.08)	(0.73,0.66,0.18)
illumination vector:		
- computed by alg.		(0.81,0.56,0.19)
- independent meas.		(0.73,0.66,0.19)

Table 9-3: Body and surface reflection vectors of the three plastic cups under yellow light

A COLOR METRIC FOR COMPUTER VISION

Glenn Healey and Thomas O. Binford

Robotics Laboratory
Computer Science Department
Stanford University
Stanford, CA 94305

Abstract

In this paper, we develop a color space metric which is useful for computer vision. While there is no shortage of proposed methods for quantifying how the human eye perceives color differences, these perceptual metrics are inherently inappropriate for computer vision. The metric we develop is appropriate for images sensed using color filters (e.g. R, G, B) as is often done in computer vision. Our approach is more general than previous approaches in that our analysis makes use of the spectral characteristics of the filters and camera and accounts for their noise properties. Our color distance function is derived as an estimate of the physical distance between normalized spectral power distributions. Components of this distance are weighted to account for sensor noise properties. A color metric has several possible uses in a computer vision system. A color metric is useful for detecting color edges, classifying intensity edges, and estimating color variation within image regions. We demonstrate the usefulness of our color metric by evaluating its performance on regions of real images.

1 Introduction

Until recently, most work in computer vision has dealt exclusively with images obtained using a single sensor. It is perhaps for this reason that many of the available algorithms for color images are straightforward multidimensional generalizations of algorithms originally intended for intensity images. Many of these generalized algorithms have been effectively applied to real images, e.g. [12]. It is reasonable to expect, however, that better uses for color information can be obtained by first examining the physics that is specific to the formation of color images. Only then can we objectively say whether color images contain useful information which is not present in intensity images and, if so, how this information can be reliably extracted.

Perhaps the most important advantage of having color information in addition to image irradiance information is that the imaged "color" of a surface of one material is more stable under changes in geometry than the corresponding image irradiance values. This assumes, of course, that some mechanism is available for factoring the intensity component out of a color signal, so that it makes sense to talk about independent intensity and color attributes of light. Shafer captured the essence of this relationship in his dichromatic reflection model [15], which decomposes both the specular and body components of reflected light into an "intensity factor" which depends on geometry but is independent of wavelength and a "spectral factor" which depends on wavelength but is independent of geometry. Although physically there are small dependencies of spectral reflectance on geometry [5], the dichromatic reflection model is a very useful approximation and has led to some impressive results in recovering intrinsic images which separate highlights from body reflection [6]. Thus, given that the "color" of an imaged surface is more stable than its image irradiance values, it is worthwhile in computer vision to study the properties of color in detail.

In this paper, we develop a color metric for computer vision. This metric is intended to quantify the significance of a color change in an image. Much work has been done in an attempt to quantify perceptual color differences [18]. Unfortunately, there does not exist an obvious relationship between the way color is perceived by a human and the way color is sensed by a computer vision system. Some work in computer vision has been done on developing color metrics for the purpose of detecting color edges [10], [11], [13]. None of the proposed metrics, however, consider the underlying properties of the sensors being used. In this work, we develop a more general color metric which accounts for the spectral properties of the camera and filters and their noise characteristics.

We do not believe that color edge detection is a particularly useful step to be taken in computer vision. The experiments of Novak and Shafer [11] and Nevatia [10] have demonstrated that a large majority of detected color edges are also detected as irradiance edges. Interestingly, human subjects have been unable to bring color edges into focus when the color edge does not coincide with a brightness edge [17].

We intend to apply our color metric to certain regions of an image following the detection of image irradiance edges. This is particularly useful for classifying irradiance edges during physical segmentation as discussed in [5]. For example, given an object of a single material which is illuminated by a single spectral power distribution, there will be an irradiance edge, but not a "color" edge across a surface orientation discontinuity. Similarly, a specular irradiance edge in an image will indicate the presence of an inhomogeneous material if there is a corresponding "color" edge [5]. On the other hand, a specular irradiance edge without a "color" edge indicates the presence of a homogeneous material. Our color metric is also useful for estimating the color variation within image regions of continuous image irradiance. This variation has a strong relationship to the material composition of an object's surface. We hope to be able to use this measure for object recognition.

We begin in section 2 with a description of color sensing. We show that given a finite number of color filters it is possible to recover a finite dimensional approximation to a color signal. In section 3, we introduce our color normalization procedure and develop a metric space for continuous color signals. The analysis of section 3 is specialized to a finite dimensional representation in section 4. The effects of noise on our metric space are considered in section 5. Section 6 presents experimental results.

2 Color Sensing

The image sensors considered in this work measure the irradiance of incident light across a plane. The properties of the light impinging on the sensor plane can be characterized by the function $I(x, y, \lambda)$ where I is irradiance, x and y are spatial coordinates in the image plane, and λ is wavelength. Information about the spectral properties of the incident light can be obtained by using several sensors (e.g. color filters) with differing wavelength responses. For the remainder of this section, we consider a fixed image location (x_0, y_0) and abbreviate $I(x_0, y_0, \lambda)$ by $I(\lambda)$.

It has been standard in computer vision to digitize an image of RGB values using "red", "green", and "blue" filters and to refer to these RGB values as if they represent something fundamental. In reality, there are an infinite number of combinations of filters and cameras that might be used to obtain RGB values. In general, different "red" filters placed in front of different CCD cameras will give us different R values for the same incident light. In this work, we consider color descriptions that are related to the physical properties of the incident light and that are therefore somewhat less dependent on the actual sensors used.

We now describe a method for recovering an approximation to the function $I(\lambda)$ at a point in an image. At each image point, we measure the outputs s_i of n sensors. Each sensor has a certain spectral response which we denote by the function $f_i(\lambda)$. In a typical system, each function $f_i(\lambda)$ will correspond to the product of a color filter transmission function with the spectral response of the camera. Therefore, at each image point we have the measured values s_i $(0 \leq i \leq n-1)$ given by

$$s_i = \int_\lambda f_i(\lambda) I(\lambda) d\lambda \qquad (2.1)$$

where λ ranges over the entire electromagnetic spectrum. In this work, we restrict our interest in the behavior of $I(\lambda)$ to the visible region of the spectrum. Therefore, the functions $f_i(\lambda)$ will be nonzero only for values of λ in the visible range. We denote the visible range by $[\lambda_1, \lambda_2]$. Since many cameras respond to light in the near infrared region of the spectrum, it may be necessary to use an IR cutoff filter to block light for which $\lambda > \lambda_2$.

Suppose we approximate the function $I(\lambda)$ by a linear combination of m basis functions $b_j(\lambda)$. If we let a_j be the components of $I(\lambda)$ on this basis, then we have

$$I(\lambda) \approx \sum_{0 \leq j \leq m-1} a_j b_j(\lambda) \qquad (2.2).$$

Substituting, (2.1) becomes

$$s_i = \int_{\lambda_1}^{\lambda_2} f_i(\lambda) \Big(\sum_{0 \leq j \leq m-1} a_j b_j(\lambda) \Big) d\lambda \qquad (2.3)$$

which may be written

$$s_i = \sum_{0 \leq j \leq m-1} a_j \left(\int_{\lambda_1}^{\lambda_2} f_i(\lambda) b_j(\lambda) d\lambda \right) \qquad (2.4)$$

Let

$$k_{ij} = \int_{\lambda_1}^{\lambda_2} f_i(\lambda) b_j(\lambda) d\lambda \qquad (2.5)$$

denote the integral in (2.4). Then k_{ij} is a constant which depends only on the ith filter function and the jth basis function. Let S be the n-dimensional vector defined by $S = [s_0, s_1, \ldots, s_{n-1}]^T$. Let K be the n x m constant matrix defined by $K(i,j) = k_{ij}$. Let A be the m-dimensional vector defined by $A = [a_0, a_1, \ldots, a_{m-1}]^T$. Then we have the linear system of n equations

$$S = KA \qquad (2.6).$$

If we choose our filters $f_i(\lambda)$ and basis functions $b_j(\lambda)$ such that K has maximal rank, then the n sensor outputs $s_0, s_1, ..., s_{n-1}$ uniquely determine n components $a_0, a_1, ..., a_{n-1}$ of $I(\lambda)$. Therefore, by letting m=n in (2.6) we can recover an approximation to the function $I(\lambda)$ given by

$$\widetilde{I(\lambda)} = B^T(\lambda) A \qquad (2.7a)$$

where

$$A = K^{-1} S \qquad (2.7b)$$

and $B(\lambda)$ is the vector $[b_0(\lambda), b_1(\lambda), \ldots, b_{n-1}(\lambda)]^T$.

In summary, $\widetilde{I(\lambda)}$ is the unique approximation to $I(\lambda)$ which lies in the space spanned by the n basis functions $b_0(\lambda), b_1(\lambda), \ldots, b_{n-1}(\lambda)$ and which also satisfies the n constraints of (2.1).

There are two sources of error in the approximation $\widetilde{I(\lambda)}$. The first source of error is noisy sensor measurements S. The second source of error is the possibility that $I(\lambda)$ cannot be exactly represented in the space spanned by the basis functions $b_0(\lambda), b_1(\lambda), \ldots, b_{n-1}(\lambda)$. The effects of sensor noise are discussed in section 5. A discussion of approximation error is not central to the development of this work, but is included for completeness in an appendix.

The color recovery method described in this section has frequently been confused with techniques that express surface reflectance as a linear combination of basis functions. We believe that expressing surface reflectance using such a linear model was first done by Sallstrom [14] and subsequently used by many others, e.g. [2], [3], [7]. *In this section, we are doing something which is fundamentally different.* We express the *function* $I(\lambda)$ (*not the surface reflectance*) as a linear combination of basis functions. For now, we are interested in an approximation to $I(\lambda)$ and not in an approximation to surface reflectance.

3 A Metric Space for Normalized Color

In this section, we develop a normalization procedure for color. This normalization is motivated by the physics of color image formation and preserves what we feel are the most important properties of color. After defining the normalization procedure, we define a distance function on the space of normalized colors. We note that the analysis of this section assumes the recovery of a continuous function $I(\lambda)$ using noiseless sensors. In section 4 we specialize the analysis to finite dimensional approximations like those recovered using the technique of section 2. In section 5 we consider the effects of sensor noise.

We begin by distinguishing the total power of the function $I(\lambda)$ from its color. The total power of $I(\lambda)$ is given by the $L^1[\lambda_1, \lambda_2]$ norm

$$\|I(\lambda)\|_1 = \int_{\lambda_1}^{\lambda_2} I(\lambda) d\lambda \qquad (3.1).$$

We define the normalized physical color of $I(\lambda)$ by the function $\widehat{I(\lambda)}$ having unit total power

$$\widehat{I(\lambda)} = \frac{I(\lambda)}{\|I(\lambda)\|_1} \qquad (3.2)$$

The space of normalized physical colors is the space of all continuous nonnegative functions $\widehat{I(\lambda)}$ on $[\lambda_1, \lambda_2]$ having unit total power.

In [5] it is shown that to a very good approximation spectral reflectance is independent of geometry. If we assume this independence and a technique for removing highlights such as [6], then we get the desirable property that for a surface of a single material illuminated by a single spectral power distribution the normalized physical color of all image points corresponding to that surface will be identical. In this sense, (3.2) serves to factor out geometric information and preserve information about source color and surface spectral reflectance.

Given any two normalized physical colors $\widehat{I_1(\lambda)}$ and $\widehat{I_2(\lambda)}$ we define the distance from $\widehat{I_1(\lambda)}$ to $\widehat{I_2(\lambda)}$ by the $L^2[\lambda_1, \lambda_2]$ distance function

$$d(\widehat{I_1(\lambda)}, \widehat{I_2(\lambda)}) = \sqrt{\int_{\lambda_1}^{\lambda_2} \left[\widehat{I_1(\lambda)} - \widehat{I_2(\lambda)}\right]^2 d\lambda} \qquad (3.3)$$

For reasons relevant to our application, we use the $L^2[\lambda_1, \lambda_2]$ distance of (3.3) rather than the commonly used maximum distance defined by

$$d_{max}(\widehat{I_1(\lambda)}, \widehat{I_2(\lambda)}) = \max_{\lambda_1 \leq \lambda \leq \lambda_2} |\widehat{I_1(\lambda)} - \widehat{I_2(\lambda)}| \quad (3.4).$$

While inaccuracies over a small range of λ can cause large deviations of d_{max}, the distance d of (3.3) depends on the distance between the functions integrated over the entire spectrum. Therefore, the $L^2[\lambda_1, \lambda_2]$ distance will usually give a more reliable characterization of the distance between two measured functions than the distance d_{max}.

Although we have not motivated our decision to compute distances between functions normalized in $L^1[\lambda_1, \lambda_2]$ using the $L^2[\lambda_1, \lambda_2]$ metric, functions treated in this way have several useful properties. These properties are discussed in [5].

4 A Finite Dimensional Representation

The functions $I(\lambda)$ in the discussion of section 3 were assumed to be arbitrary nonnegative continuous functions on $[\lambda_1, \lambda_2]$, i.e. nonnegative elements of the space $C[\lambda_1, \lambda_2]$. Using the color recovery method of section 2, however, we are only able to recover finite dimensional approximations to these functions. In this section, we specialize the analysis of section 3 to finite dimensional approximations which can be recovered using the method of section 2. Figure 1 depicts the different levels of color representation in our system.

In general, we begin by restricting ourselves to some n-dimensional subset S of $C[\lambda_1, \lambda_2]$. If S is specified by a set of basis functions, then the Gram-Schmidt process [4] can be used to construct an orthonormal basis for S. This orthonormal basis may then be taken to be the basis $B(\lambda)$ of section 2.

For the purpose of concreteness, we will take S to be the set of polynomials of degree n-1 and the orthonormal basis to be the first n normalized Legendre polynomials. The normalized Legendre polynomials are given by

$$p_i(\lambda) = \sqrt{\frac{2i+1}{2}} P_i(\lambda) \qquad (4.1)$$

where $P_i(\lambda)$ is the Legendre polynomial of degree i. The functions $p_i(\lambda)$ are orthonormal in the sense that

$$\int_{\lambda_1}^{\lambda_2} p_i(\lambda) p_j(\lambda) = \begin{cases} 1, & \text{if } i = j; \\ 0, & \text{if } i \neq j. \end{cases} \qquad (4.2)$$

We note that the analysis of this section applies more generally to any orthonormal basis for which one of the basis functions is a constant.

4.1 Computing Normalized Physical Color $\widehat{I(\lambda)}$

Using the normalized Legendre polynomial representation, the technique of section 2 recovers an approximation to $I(\lambda)$ of the form

$$I(\lambda) = \sum_{0 \leq i \leq m-1} a_i p_i(\lambda) \qquad (4.3)$$

The total power of $I(\lambda)$ is given by

$$\|I(\lambda)\|_1 = \int_{\lambda_1}^{\lambda_2} I(\lambda) d\lambda = \int_{\lambda_1}^{\lambda_2} \Big[\sum_{0 \leq i \leq n-1} a_i p_i(\lambda) \Big] d\lambda = \sqrt{2} a_0 \qquad (4.4)$$

where the last step follows from the orthogonality of the functions $p_i(\lambda)$. Therefore, the total power of $I(\lambda)$ may be determined by only considering the first coefficient in the normalized Legendre polynomial representation. From the total power of $I(\lambda)$, we can determine the normalized physical color $\widehat{I(\lambda)}$ by

$$\widehat{I(\lambda)} = \frac{I(\lambda)}{\sqrt{2} a_0} = \sum_{0 \leq i \leq n-1} \acute{a}_i p_i(\lambda) \qquad (4.5)$$

where

$$\acute{a}_i = \frac{a_i}{\sqrt{2} a_0} \qquad (4.6)$$

4.2 Metric Space Properties

For two functions $I_1(\lambda)$ and $I_2(\lambda)$ given by

$$I_1(\lambda) = \sum_{0 \leq i \leq n-1} r_i p_i(\lambda), \qquad I_2(\lambda) = \sum_{0 \leq i \leq n-1} s_i p_i(\lambda) \qquad (4.7)$$

the normalized physical colors are

$$\widehat{I_1(\lambda)} = \sum_{0 \leq i \leq n-1} r_i p_i(\lambda), \qquad \widehat{I_2(\lambda)} = \sum_{0 \leq i \leq n-1} s_i p_i(\lambda) \qquad (4.8)$$

where the r_i and s_i are computed as in (4.6).

The color space distance between $\widehat{I_1(\lambda)}$ and $\widehat{I_2(\lambda)}$ is given by

$$d(\widehat{I_1(\lambda)}, \widehat{I_2(\lambda)}) = \sqrt{\int_{\lambda_1}^{\lambda_2} \Big[\sum_{0 \leq i \leq n-1} (\hat{r}_i - \hat{s}_i) p_i(\lambda) \Big]^2 d\lambda} \tag{4.9}$$

which simplifies because of orthogonality to

$$d(\widehat{I_1(\lambda)}, \widehat{I_2(\lambda)}) = \sqrt{\sum_{0 \leq i \leq n-1} (\hat{r}_i - \hat{s}_i)^2} \tag{4.10}$$

4.3 Color Space as a Subset of R^{n-1}

From (4.5), our representation for normalized physical colors is

$$\widehat{I(\lambda)} = \sum_{0 \leq i \leq n-1} \hat{a}_i p_i(\lambda) \qquad (4.11).$$

From (4.6) we see that all normalized physical colors have $\hat{a}_0 = 1/\sqrt{2}$. We can write

$$\widehat{I(\lambda)} = \frac{1}{2} + \sum_{1 \leq i \leq n-1} \hat{a}_i p_i(\lambda) \qquad (4.12).$$

We can consider the normalized physical color $\widehat{I(\lambda)}$ to be the point in R^{n-1} with coordinates $\hat{a}_1$,$\hat{a}_2$,...,$\hat{a}_{n-1}$. Since we require $\widehat{I(\lambda)} \geq 0$ on $\lambda_1 \leq \lambda \leq \lambda_2$, normalized physical colors form a proper subset of R^{n-1}. To be precise, normalized physical colors are those points of R^{n-1} for which

$$\sum_{1 \leq i \leq n-1} \hat{a}_i p_i(\lambda) \geq -\frac{1}{2} \qquad -1 \leq \lambda \leq 1 \qquad (4.13).$$

Let C^{n-1} be the set of points in R^{n-1} which are normalized physical colors. Points in C^{n-1} will be referred to as

$$A = (\hat{a}_1, \hat{a}_2, \ldots, \hat{a}_{n-1}) \tag{4.14}$$

where the coordinates $\hat{a}_1$, $\hat{a}_2$, ..., $\hat{a}_{n-1}$ have the same significance as in (4.12).

An important consequence of viewing colors as points in C^{n-1} is that the usual euclidean metric in R^{n-1} is equivalent to the metric we defined for normalized physical colors in section 3. This is seen by examining (4.10) and recalling that $\hat{r}_0 = \hat{s}_0$. Thus, our representation in terms of normalized Legendre polynomials allows intuition about the familiar distance in R^{n-1} to be applied to distance in color space.

5 Modeling Sensor Noise

If we are able to make perfect measurements $s_0, s_1, \ldots, s_{n-1}$, then (2.7) allows us to compute the approximation to $I(\lambda)$ given by

$$\widetilde{I(\lambda)} = B^T(\lambda) A \tag{5.1}$$

In real physical situations, each of the measurements $s_0, s_1, \ldots, s_{n-1}$ will have some amount of variability. In this section, we examine how the variance in the measurements S affects the computed approximation of (5.1).

Let $\overline{A}$ denote $E(A)$ and $\overline{S}$ denote E(S). From (2.7b), we have

$$\overline{A} = E[K^{-1}S] = K^{-1}\overline{S} \tag{5.2}$$

Let Σ_A be the n x n covariance matrix of A defined by

$$\begin{aligned} \Sigma_A \qquad &= E[(A -- \overline{A})(A - \overline{A})^T] && (5.3) \\ &= E[(K^{-1}S - K^{-1}\overline{S})(K^{-1}S - K^{-1}\overline{S})^T] && (5.4) \\ &= K^{-1}E[(S - \overline{S})(S - \overline{S})^T](K^{-1})^T && (5.5) \end{aligned}$$

which may be written

$$\Sigma_A = K^{-1}\Sigma_S(K^{-1})^T \tag{5.6}$$

where Σ_S is the n x n covariance matrix of S.

For most real sensors, it is reasonable to assume that A has the multivariate normal density given by

$$p(A) = \frac{1}{(2\pi)^{n/2} ||\Sigma_A||^{1/2}} e^{-\frac{1}{2}(A - \overline{A})^T \Sigma_A^{-1} (A - \overline{A})} \qquad (5.7).$$

Contours of constant density are hyperellipsoids satisfying

$$(A - \overline{A})^T \Sigma_A^{-1} (A - \overline{A}) = C \tag{5.8}$$

where C is a constant. Thus, from Σ_A we can determine the scatter of the data in any direction. It follows that, in general, points in C^{n-1} will tend to exhibit greater dispersion in certain directions than in others. Thus the euclidean metric for C^{n-1} which is appropriate for noiseless measurements (see section 4) will be inappropriate for real sensor measurements. This is because directions in C^{n-1} along which sensor noise is amplified can dominate the euclidean distance of (4.10). It is desirable to develop a color metric which can take into account this anisotropic property of C^{n-1} and thereby produce a distance between two normalized physical colors which is relatively independent of scatter due to noise. We propose as a metric the Mahalanobis distance [16] given by

$$d(A_1, A_2) = \sqrt{(A_1 - A_2)^T \Sigma_A^{-1} (A_1 - A_2)} \qquad (5.9)$$

where A_1 and A_2 are points in C^{n-1}. The metric of (5.9) has the effect of normalizing by the variance in each direction, thus giving a more representative estimate of the physical distance between A_1 and A_2 than the euclidean distance. We observe that if Σ_A is unitary, then C^{n-1} is isotropic. As expected, for this case (5.9) simplifies to the scaled euclidean distance.

6 Experiments

In this section we show the results of some preliminary experiments using the metric of (5.9). For these experiments we digitized a color image of the valve part from the SUCCESSOR vision system project [1]. Four different sensors were used to obtain the color information. The spectral response curves for the four sensors are shown in figure 2. The large amplitude curve indicates the spectral response of the camera and the three other curves indicate the spectral response of the camera multiplied by the response of the color filters used. These curves were taken from manufacturer's specifications. In the future, we hope to be able to measure these curves directly. The basis functions used for the experiments are the normalized Legendre polynomials of section 4. The resulting matrix K^{-1} is

$$K^{-1} = \begin{pmatrix} -0.279 & 0.540 & 0.463 & 0.648 \\ -0.609 & 0.964 & 0.996 & 0.694 \\ -0.507 & 0.751 & 0.578 & 1.082 \\ -0.446 & 0.531 & 0.892 & 0.568 \end{pmatrix}$$

Here we will show the use of our color metric in detecting material edges. Figure 3 shows a scatterplot of points in a 2-D cross section of C^3 across an edge corresponding to two different metals. P1 indicates the strength of the component corresponding to $p_1(\lambda)$ and P2 indicates the strength of the component corresponding to $p_2(\lambda)$ using the color recovery method of section 2. The points lie approximately in two clusters corresponding to the two metals, with some blurring across the edge. Figure 4 shows the result of applying our color metric point by point to a line of pixels across the material edge. The large central maximum indicates the presence of a color edge which in this case corresponds to a material edge. Figures 5 and 6 show similar results for another material edge in the same image.

Since intensity information has been factored out, the large central maxima in figures 4 and 6 indicate color discontinuities. We note that it is not necessary to localize edges precisely using color since many techniques are available for localizing the corresponding intensity edges, e.g. [9]. In the future, we hope to apply our metric to estimating color variation within image regions of continuous irradiance.

Appendix: Sensor Selection and Approximation Error

Given the recovery technique described in section 2, we examine how the choice of the sensors $f_i(\lambda)$ can affect the quality of the recovered approximation to $I(\lambda)$. In particular, we show that given an orthonormal system of n basis functions, it is possible to choose n sensors such that the recovered approximation $\widetilde{I(\lambda)}$ is the best least squares approximation to $I(\lambda)$ in the space spanned by the basis.

More formally, let $\phi_0(\lambda), \phi_1(\lambda), \ldots \phi_{n-1}(\lambda)$ be an orthonormal set of basis functions in $L^2[\lambda_1, \lambda_2]$. Then given n sensor measurements S, the procedure of section 2 allows us to recover an approximation $\widetilde{I(\lambda)}$ of the form

$$\widetilde{I(\lambda)} = \sum_{0 \leq i \leq n-1} d_i \phi_i(\lambda) \qquad (A.1).$$

The best approximation $\widetilde{I(\lambda)}$ in the least squares sense is the choice of the d_i which minimizes

$$M = \int_{\lambda_1}^{\lambda_2} \left[I(\lambda) - \sum_{0 \leq i \leq n-1} d_i \lambda^i \right]^2 d\lambda \qquad (A.2).$$

We observe that (2.1) may be written

$$s_i = \langle f_i, I \rangle \qquad (A.3)$$

where $\langle \cdot, \cdot \rangle$ denotes the $L^2[\lambda_1, \lambda_2]$ inner product. Thus, if we define

$$f_i(\lambda) = \phi_i(\lambda) \quad i = 0, 1, \ldots, n-1 \qquad (A.4)$$

then we will have

$$s_i = \langle \phi_i, I \rangle \qquad (A.5).$$

Since the $\phi_i(\lambda)$ are orthonormal functions, K will be the identity matrix. From (2.7), the technique of section 2 will recover the approximation $\widetilde{I(\lambda)}^*$ given by

$$\widetilde{I(\lambda)}^* = \sum_{0 \leq i \leq n-1} \langle \phi_i, I \rangle \phi_i(\lambda) \qquad (A.6).$$

It is easy to show [4] that the approximation $\widetilde{I(\lambda)}^*$ given by (A.6) minimizes M and that the error is given by

$$||I(\lambda) - \widetilde{I(\lambda)}^*||^2 = ||I(\lambda)||^2 - ||\widetilde{I(\lambda)}^*||^2 \qquad (A.7)$$

Therefore, it is possible to choose sensors $f_i(\lambda)$ such that the procedure of section 5 computes the best least squares approximation to an arbitrary function $I(\lambda)$.

It is natural to ask how large n should be in order that the function $\widetilde{I(\lambda)}$ is a good approximation to $I(\lambda)$. This depends, of course, on the properties of the function $I(\lambda)$ and how well these properties are captured by the basis functions $B(\lambda)$. Maloney [8] has done a thorough analysis of reflectance functions $R(\lambda)$ for real materials by considering both empirical data and the fundamental physical processes which determine the properties of $R(\lambda)$. He has concluded that at least five basis functions are required to accurately characterize $R(\lambda)$. *Unfortunately, no corresponding analysis* has been done for illuminants.

Acknowledgements

Support for this work was provided in part by the Air Force Office of Scientific Research under contract F33615-85-C-5106 and by the Advanced Research Projects Agency of the Department of Defense under Knowledge Based Vision ADS subcontract S1093 S-1.

References

[1] Binford, T.O. and T. Levitt and W. Mann, "Bayesian Inference in Model-Based Machine Vision," Proceedings of the Workshop on Uncertainty in Artificial Intelligence, 1987.

[2] Brill, M., "A Device Performing Illuminant-Invariant Assessment of Chromatic Relations", *Journal Theoretical Biology*, 1978, 71, 473-478.

[3] Buchsbaum, G. "A Spatial Processor Model for Object Colour Perception," *Journal Franklin Institute*, 1980, 310, 1-26.

[4] Courant, R. and D. Hilbert, *Methods of Mathematical Physics*, Volume 1, Wiley & Sons, New York, 1953.

[5] Healey, G. and T.O. Binford, "The Role and Use of Color in a General Vision system," Proceedings of the ARPA Image Understanding Workshop, USC, 1987.

[6] Klinker, G. and S. Shafer and T. Kanade, "Using a Color Reflection Model to Separate Highlights from Object Color," Proceedings of the First International Conference on Computer Vision (London: June 1987), p. 145-150.

[7] Maloney, L. and Wandell, B., "Color Constancy: A Method for Recovering Surface Spectral Reflectance," *Journal of the Optical Society of America A*, VOl. 3, October 1986, 1673-1683.

[8] Maloney, L., " Evaluation of Linear Models of Surface Spectral Reflectance with Small Numbers of Parameters," *Journal of the Optical Society of America A*, Vol. 3, October 1986, 1673-1683.

[9] Nalwa, V. "Edge-Detector Resolution Improvement by Image Interpolation," *IEEE Transactions on PAMI*, 9, No. 3, May 1987.

[10] Nevatia, R. "A Color Edge Detector and Its Use in Scene Segmentation," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-7, no. 11, 1977, pp. 820-826.

[11] Novak, C. and S. Shafer, "Color Edge Detection" in "Image Understanding Research at CMU" by Takeo Kanade, Proceedings of ARPA Image Understanding Workshop, USC, 1987, pp. 35-37.

[12] Ohlander, R., and K. Price, and D.R. Reddy, "Picture Segmentation Using a Recursive Region Splitting Method," *Computer Graphics and Image Processing*, 8:313-333, 1978.

[13] Robinson, G.S., "Color Edge Detection," *Optical Engineering*, vol. 16, no. 5, September-October, 1977, pp. 479-484.

[14] Sallstrom, P. "Colour and Physics; Some Remarks Concerning the Physical Aspects of Human Colour Vision," University of Stockholm: Institute of Physics Report 73-09, 1973.

[15] Shafer, S., "Using Color to Separate Reflection Components," University of Rochester TR 136, April 1984.

[16] Simon, J.C., *Patterns and Operators*, McGraw-Hill, New York, 1986.

[17] Wolfe, J. "*Hidden Visual Processes*," *Scientific American*, February, 1983, pp. 94-103.

[18] Wyszecki, G. and W. Stiles, *Color Science*, John Wiley & Sons, Inc., New York, 1967.

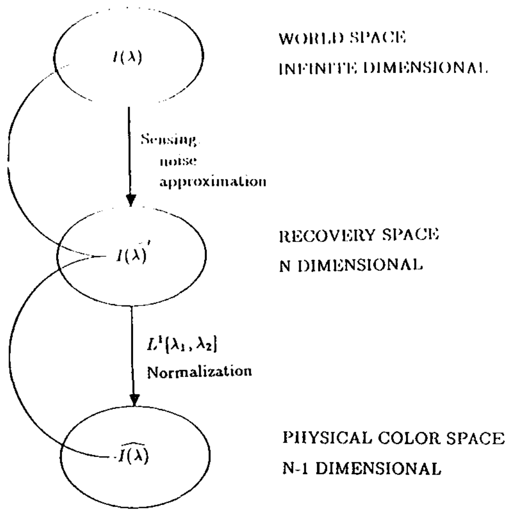


Figure 1

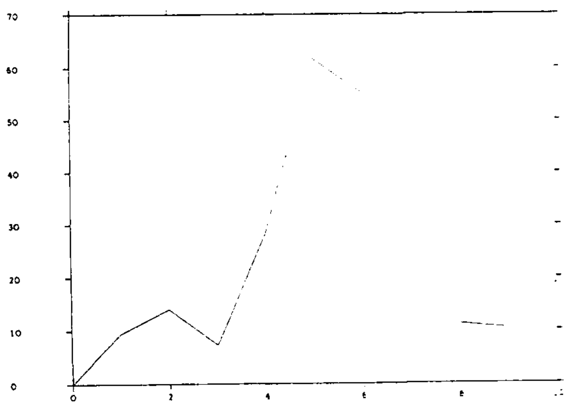


Figure 4

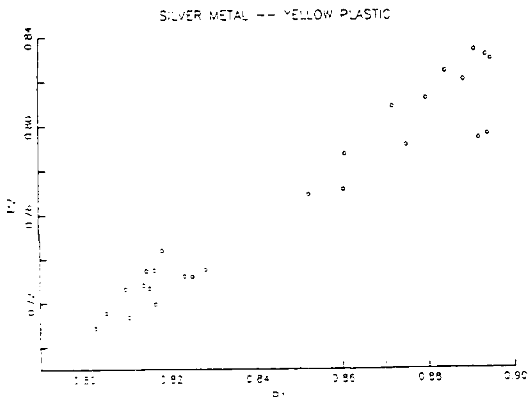


Figure 5

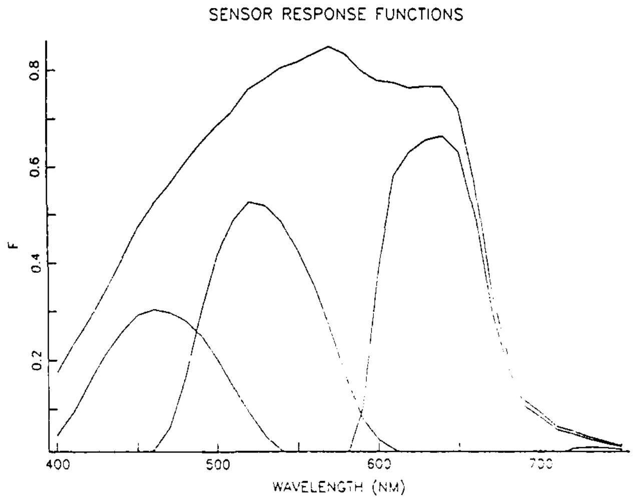


Figure 2

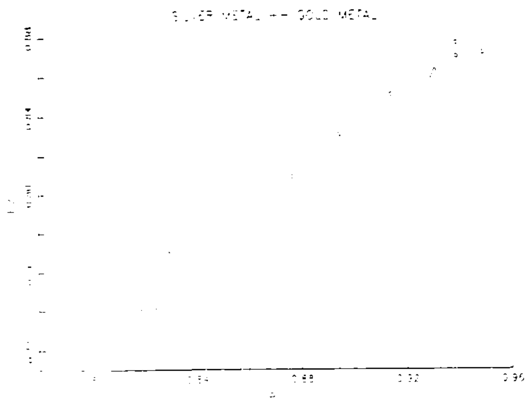


Figure 3

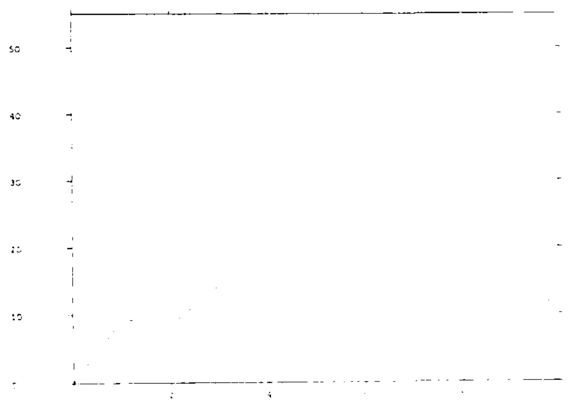


Figure 6

COMBINING INFORMATION IN LOW-LEVEL VISION

John (Yiannis) Aloimonos
Anup Basu

Computer Vision Laboratory
Center for Automation Research
University of Maryland
College Park, MD 20742-3411

ABSTRACT

Low level modern computer vision is not domain dependent, but concentrates on problems that correspond to identifiable modules in the human visual system. Several theories have been proposed in the literature for the computation of shape from shading, shape from texture, retinal motion from spatiotemporal derivatives of the image intensity function and the like.

The problems with some of the existing approaches are basically the following:

(1) The employed assumptions are usually very strong (they are not present in a large subset of real images), and so some of the algorithms fail when applied to real images.

(2) Usually the constraints from the geometry and the physics of the problem are not enough to guarantee uniqueness of the computed parameters. In this case, strong additional assumptions about the world are used, in order to restrict the space of all solutions to a unique value.

(3) Even if no assumptions at all are used and the physical constraints are enough to guarantee uniqueness of the computed parameters, then in most cases the resulting algorithms are not robust, in the sense that if there is a slight error in the input (i.e. a small amount of noise in the image), this results in a catastrophic error in the output (computed parameters), and this is observed from experiments.

It turns out that if several available cues are combined, then the above mentioned problems disappear in most cases; the resulting algorithms compute robustly and uniquely the intrinsic parameters (shape, depth, motion, etc.).

In this paper the problem of machine vision is explored from its basics. A low level mathematical theory is presented for the unique and robust computation of intrinsic parameters. The computational aspect of the theory envisages a cooperative highly parallel implementation, bringing in information from five different sources (shading, texture, motion, contour and stereo), to resolve ambiguities and ensure uniqueness of the intrinsic parameters.

1. INTRODUCTION

A large part of Low and Intermediate Level Vision, i.e. problems such as *shape from texture, shape from shading, structure from motion, three-dimensional motion analysis* and *shape from contour, depth computation and determination of light source direction*, are studied. All these problems have been studied in the past, using very few information sources. It was proved that in order to achieve uniqueness of the underlying computations, some assumptions should be employed. This research discovered for the first time constraints relating three-dimensional information with two-dimensional image properties and resulted in algorithms that worked in synthetic laboratory images and in some restricted domain of natural images. In this work, we study the above problems, by using as much information as is available from different cues (stereo, motion, contour, shading and texture). This results in algorithms that provably compute uniquely what they are supposed to compute.

The basic ideas in this paper are centered around the fact that minimal assumptions, uniqueness and stability, should (and can) be the first and basic requirements for a visual computation.

The next section introduces the reader to basic problems in computer vision and establishes some of the required nomenclature, and Section 3 reviews a large part of previous work, gives the motivation for the research needed and describes the results we have obtained.

2. PREREQUISITES

2.1. The Goal of Machine Vision

It is very difficult to define the central problem of computer vision or vision in general, as in many other scientific fields. What goes on inside our heads when we see? Most people take seeing so much for granted that few will ever have considered the question seriously. Here we attempt to give the following loose definition of the central problem of computer vision:

> *"The central problem of computer vision is: from one or a sequence of images of a moving or stationary object or a scene, taken by a monocular (one eye) or polyaocular (many eyes) moving o- stationary observer, to understand the object or the scene and its three-dimensional properties."*

The support of the Defense Advanced Research Projects Agency and the U.S. Army Center for Night Vision and Electro-Optics under Contract DAAB07-86-K-F073 is gratefully acknowledged, as is the help of Sandy German and Barbara Bull in preparing this paper.

The reader will immediately observe that all the terms in the above definition are well defined, with the exception of the term "understand". What is really the meaning of "understand" with respect to this problem? The problem of finding meaning is the central one in artificial intelligence and it is by no means answered. For this reason, because various researchers understand meaning in different ways, there have basically been two schools of thought in computer vision. Although no clear distinction between them can be made, we can safely differentiate them into two schools: *Reconstruction* and *Recognition*. The reconstruction school worries about the reconstruction of the physical parameters of the visual world, such as the depth or orientation of surfaces, the boundaries of objects, the direction of light sources and the like. The recognition school worries about the recognition or description of objects that we see and involves processes whose end product is some piece of behavior like a decision or a motion. Both schools have strong ties with psychology and neuroscience and it is strongly believed at this point that both schools will merge into a new one that will, it is hoped, find an answer to the difficult questions of the vision problem. Figure 2.1 shows a schematic description of the above mentioned schools of thought.

2.2. The Machine Vision Goal Revisited

Up to this point we have been rather general since we have been talking about computer vision as having as its goal the development of a universal visual system. We consider the explanation of Horn [1986] very much adequate and we adopt it here. To be more specific, a machine vision system analyzes images and produces descriptions of what is imaged. These descriptions must capture the aspects of the objects being imaged that are useful in carrying out some task. So, we consider the machine vision system as part of a larger entity that interacts with the environment. The vision system can be considered an element of a feedback loop that is concerned with sensing, while other elements are dedicated to decision making and the implementation of these decisions. The input to the machine vision system is an image, or several images, while its output is a description that should satisfy at least the following two criteria (following Horn [1986]):

a) *It must bear a relevant relationship to what is being imaged*;

b) *It must contain all the information needed for the specific task.*

Obviously, the first criterion ensures that the description depends in some way on the visual input. The second ensures that the information provided is useful. Something has to be said about the concept of description that we used above. An object does not have a unique description. We can think of descriptions at many levels of detail and from many points of view. Fortunately, we can avoid this potential philosophical snare by considering the task for which the description is intended. That is, we do not want just any description of

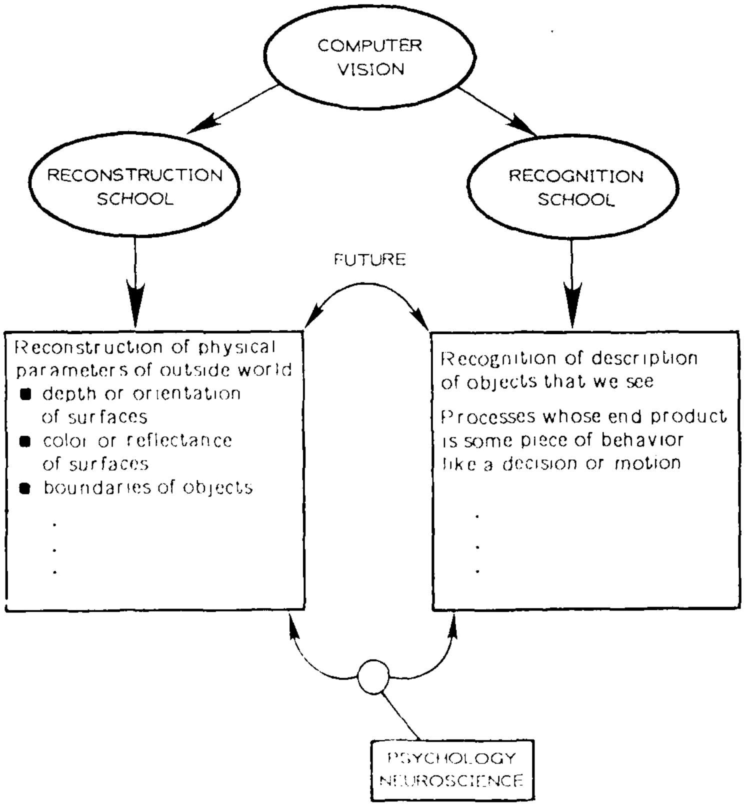

COMPUTER VISION
RECONSTRUCTION SCHOOL
RECOGNITION SCHOOL
FUTURE
Reconstruction of physical parameters of outside world
■ depth or orientation of surfaces
■ color or reflectance of surfaces
■ boundaries of objects
.
.
.
Recognition of description of objects that we see
Processes whose end product is some piece of behavior like a decision or motion
.
.
.
PSYCHOLOGY NEUROSCIENCE

Figure 2.1: The two schools in computer vision.

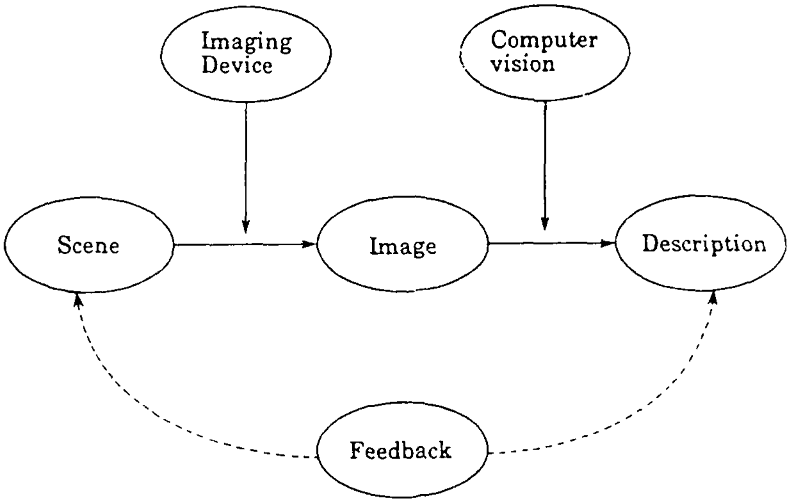


Figure 2.2: A computer vision task.

what is imaged, but one that allows us to take appropriate action.

An example, borrowed from Horn [1986], may help to clarify these ideas. Consider the task of picking up parts from a conveyor belt. The parts may be randomly oriented and positioned on the belt. There may be several different types of parts, with each to be loaded into a different fixture. The vision system is provided with images of the objects as they are transported past a camera mounted above the belt. The descriptions that the system has to produce in this case are simple. It need only give the position, orientation and type of each object. This description may be just a few numbers. In other situations an elaborate symbolic description may be needed. Figure 2.2 depicts a vision system.

2.3. Current Research Status

For a clear exposition on how the field shaped up in its current form, see [Horn, 1986]. Modern computer vision worries about concentrating on topics that correspond to identifiable modules in the human visual system. And although we don't know what exactly these modules are, we understand that there should exist modules that compute 3-D parameters form specific cues, such as shading, motion, stereo, contours and texture. When we say 3-D parameters, we mean intrinsic images, such as shape, depth, reflectance, three-dimensional motion, illuminant direction and the like. So, one could say that a large part of today's research is:

Compute Y from X.

where Y is an intrinsic property (shape, depth, retinal and three-dimensional motion, etc.) and X is a cue in the image or a property of the observer (shading, texture, stereopsis, etc.).

The following figure broadly summarizes the status of contemporary reconstructionist computer vision. On the right, we see the various cues, and on the left the intrinsic parameters. Research tries to recover from any of the cues in the right some of the intrinsic properties on the left. An arrow from box 1 to box 2 indicates that the property in box 2 is recovered from the cue in box 1. The names along the arrows represent some of the researchers who have worked on this specific recovery. More complete references can be found in the rest of the paper. At this point we have to make clear that the *intrinsic* parameters about which we are writing can basically be classified in two categories: *retinotopic* and *non-retinotopic*. The non-retinotopic ones can be divided into features (physical parameters) and objects and relations [Ballard, 1984]. The retinotopic ones (shape, depth and the like) are the ones of most interest in this paper. These parameters are spatially indexed at every image point.[1] One might say that the retinotopic parameters are the basic subject of the Reconstruction School, and the non-retinotopic ones (features) of the Recognition School. In this paper we will mostly be talking about Low-Level Vision, and so the analysis of three-dimensional shape models and transformations, as part of High-Level Vision modules, won't be treated. Finally, it has to be said that the current status, Figure 2.3, is by no means complete. Other sources of information such as color and nonplanar contours are of great importance, but we will not discuss them here.

2.4. A Word of Caution and What is to Come

In the preceding sections, we have emphasized that contemporary computer vision is worrying about the recovery of three-dimensional (world) properties from two-dimensional image properties. By no means do we imply that this is the only issue of today's research. There is a lot of excellent research on low and high level vision, object recognition and navigation. We feel that the bulk of research (from 2-D properties to 3-D properties) is important because a clear understanding of these issues will contribute a great deal to our knowledge of extrapersonal space perception, to our understanding of the cortex and to our ability to construct machines with visual sense. Simple thinking may convince us that if we ever hope to understand how the visual system

[1] But we treat the recovery of 3-D motion and light source direction, which are not retinotopic, but global parameters.

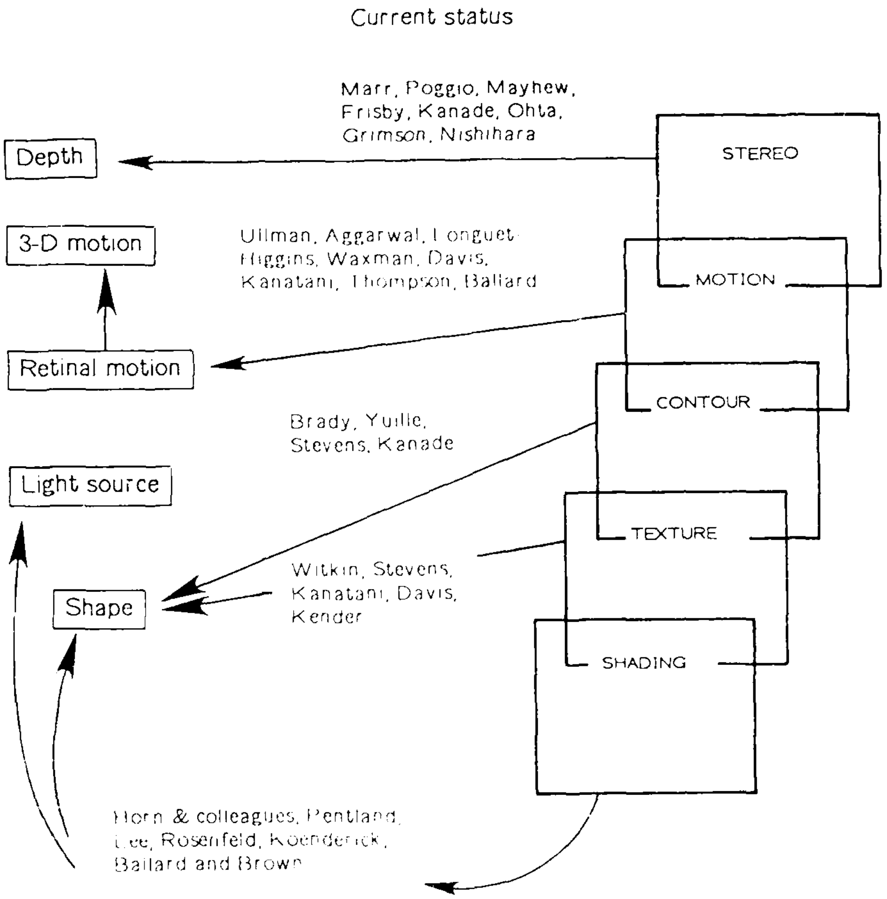

Current status
Marr, Poggio, Mayhew, Frisby, Kanade, Ohta, Grimson, Nishihara
Depth
STEREO
3-D motion
Ullman, Aggarwal, Longuet-Higgins, Waxman, Davis, Kanatani, Thompson, Ballard
MOTION
Retinal motion
CONTOUR
Brady, Yuille, Stevens, Kanade
Light source
TEXTURE
Witkin, Stevens, Kanatani, Davis, Kender
Shape
SHADING
Horn & colleagues, Pentland, Lee, Rosenfeld, Koenderick, Ballard and Brown

Figure 2.3: Current research status.

works, we must first understand that our only input is two-dimensional images, and so in order to reason about the three-dimensional world, we must discover constraints between the images and the three-dimensional world that is imaged. On the other hand, prior knowledge about the world can be of great help. We are not opposed to using *a priori* knowledge about the world in order to help the process of understanding the 3-D space from its images. But before we do that, we should first analyze the various vision problems with as few assumptions as possible, and if no solution is possible, then we should resort to additional assumptions, exactly as previous research (Fig. 2.3) discovered constraints from single cues.

3. UNIQUE INTRINSIC IMAGES: THE PROBLEM, THE ANSWER AND TECHNICAL PREREQUISITES

Here we describe what the problems of the current research status are and we propose a new approach. In the course of our analysis we give the technical prerequisites for the foundation of the technical work described later.

3.1. The Current Research Picture Revisited

Recalling the current research picture from Figure 2.3, we see that the intrinsic parameters that will be described extensively in the rest of this section are computed from some particular image cue. Indeed, *shading, texture, contours, motion* and *stereo* are very important cues for obtaining three-dimensional information, and later sections will show that. If we look carefully at the research picture from Figure 2.3, we will realize that an intrinsic parameter is computed using only a particular cue. So we have algorithms for shape from shading, shape from motion, depth from stereo, and the like. There are, however, three basic problems with this approach.

The first problem has to do with employing the right assumptions. Some of these algorithms are based on assumptions which despite their generality are not frequently present in the real world and so these algorithms fail when applied to a variety of natural images. An example of this is all the algorithms for the computation of shape from texture [Witkin, 1981; Stevens, 1979; Dunn et al., 1983]. In these algorithms the basic assumption is directional isotropy. In other words, it is assumed that contours and line segments in natural images have orientations which are uniformly distributed over all directions. Obviously, if we look around us for natural or man-made surfaces, we won't find that this assumption is always true. Other research [Aloimonos, 1986] assumed *uniform density*. Again, not all textures have this property. Of course, questions such as what is the most general assumption (i.e. the one present in most natural textures) are legitimate, but their answer may come from an empirical analysis.

The second problem has to do with uniqueness properties of the resulting algorithms. Some of the problems in Figure 2.3, as formulated, cannot have a unique solution. So, in order to bring down the space of all solutions to a unique point, assumptions are made about the world which usually are not general enough and the algorithms may fail when applied to some real images. An example of this is the shape from shading algorithms [Horn, 1977; Ikeuchi and Horn, 1981; Brooks, 1984] that use assumptions about the *smoothness of the surfaces in view.*

The third problem with the current research status is the one which has to do with the robustness or stability of the resulting algorithms. Even if theoretical analysis shows that given the constraints at hand a particular problem has a unique solution, in practice it turns out that the solution might be unstable. In other words, a very small error in the input might result in a catastrophic error in the output. An example of this is some algorithms that compute 3-D motion from retinal motion, *using only one camera* [Tsai and Huang, 1984; Longuet-Higgins and Prazdny, 1984; Prazdny, 1980, 1981]. The basic problems with the current research status can be summarized in the following table.

Table 1
Problems of Current Research Status

Problem	Example
Assumpt:ons should be employed in order to solve a problem. Usually, these assumptions are not general enough.	*Shape from texture algorithms* [Witkin, 1981; Aloimonos, 1986].
If the constraints do not guarantee uniqueness, assumptions should be employed to bring down the space of all solutions to a unique value. Usually the assumptions are not present in the world.	*Shape from shading, optic flow from image sequences* [Horn, 1986].
Even if an algorithm is proved to have a unique solution, the resulting algorithm might be unstable.	*3-D motion from optic flow* [Tsai and Huang, 1984], *image reconstruction from zero-crossings and gradients* [Hummel, 1986].

The problem of theoretical stability analysis in visual algorithms is a difficult one because distributions of the image intensity function are necessary for such a task. Whenever possible, we perform a stability analysis of the introduced algorithms, but we haven't been able to present a unified stability analysis. There is such work in progress [Aloimonos and Spetsakis, 1987]

3.2. The Regularization Paradigm

One of the best definitions of early vision is that it is the inverse of optics, i.e., a set of computational problems that both machines and biological organisms have to solve. While in classical optics the problem is to determine the images of physical objects, vision is confronted with the inverse problem of determining properties of the 3-dimensional world from the light distribution in an image, or a dynamic sequence of images. In 1923, Hadamard defined a mathematical problem to be *well-posed when its solution:*

a) exists,

b) is unique,

c) depends continuously on the initial data (is robust against noise).

Most of the problems in classical physics are well posed, and Hadamard argued that physical problems had to be well-posed. However, it seems that inverse problems are usually ill-posed. Consider, for example, the equation $y = Ax$, where A is a known operator. This equation can represent optics, where y is the image, A is the imaging process, and x is the world. So, in this case, the problem is to determine y from x. The inverse problem, i.e. find x from y, which is the problem of computer vision, is usually ill-posed when x,y belong to a Hilbert space.

Most of the early vision problem are ill-posed (shape from shading, texture, contour, optic flow from image brightness and the like). Rigorous regularization theories for solving ill-posed problems have been developed during the past years [Tichonov and Arsenin, 1977]. The basic idea of regularization techniques is to restrict the space of acceptable solutions by choosing the function that minimizes an *appropriate* functional. The regularization of the ill-posed problem of finding *x from y such that* $y = Ax$ *requires the choice of norms* $||\cdot||$ and of a stabilizing functional $||Px||$. Of course this choice is dictated by mathematical considerations and most importantly, by a physical analysis of the generic constraints of the problem. Then, several methods can be applied, as for example, find x that minimizes $||AX - y||^2 + \lambda||Px||^2$, where λ is the so-called regularization parameter, or among x that satisfy $||Px|| \leq k$, where k is a constant, find x that satisfies $||Ax - y|| =$ minimum, etc. Basically, in the regularization paradigm, a low-level vision problem that is ill-posed has to be regularized by imposing additional constraints which should be physically plausible. An example of such an additional constraint is some kind of smoothness of the unknown functions. There has been excellent research in regularizing early vision problems, originated in [Poggio and Koch, 1985]. Even though the regularization paradigm is very attractive for its mathematical elegance and for being a legitimization of already published research [Horn and Schunck, 1981; Ikeuchi and Horn, 1981; Hildreth, 1984], it has some shortcomings, in the sense that it cannot deal with the full complexity of vision. One problem is the degree of smoothness required for the unknown function that has to be recovered; for example, some unrealistic results have been reported in surface interpolation, because depth discontinuities are smoothed too much. Research on regularization in the presence of discontinuities, while *pioneering, is still premature* [Terzopoulos, 1986; Lee and Pavlidis, 1987; Shulman and Aloimonos, 1987; Nagel, 1983]. Another problem is that standard regularization theory deals with linear problems and is based on quadratic stabilizers. In the case of nonquadratic functionals standard regularization theory may be used, but the situation is problematic [Morozov, 1984]. For nonquadratic functionals, the search space may have many local minima and in this case only stochastic algorithms by Kirkpatrick described in [Poggio, 1985] might have some success. Thus, it will take the development of a rigorous theory of discontinuous regularization to justify the possibility of the regularization theory serving as a general theory of many low-level vision problems.

3.3. Results: More Information from Cooperative Sources Yields Unique and Reliable Solutions

Looking back at the current research status diagram of Figure 2.3, we see that from a particular cue a particular intrinsic property is computed. That is, no cues are combined, in most of the published work, to recover an intrinsic image. As we have already seen, this has as a result the fact that several computations do not have uniqueness properties (and so additional assumptions are needed about the world) and several computations that have uniqueness properties under *ideal conditions break down in the presence* of a small amount of noise. In order to take care of these problems, more information is needed. In particular, if we combine information from the different image cues, then several computations that did not have uniqueness properties might now have them, simply because the unknown parameters are subject to more constraints that guarantee uniqueness, and several computations which even though they had uniqueness properties were very unstable are now robust, simply because the additional constraints do not let the solution escape from its actual position. The proposed framework for the *computation of intrinsic images is given in the following Figure* 3.1. The reader should compare this with Figure 2.3 of Section 2.3 to realize that new information is combined from different cues to recover the intrinsic parameters.

It is worth noting that very recently a few researchers have realized the need for combination of information from different image cues for better estimation of intrinsic parameters. *In* particular, there is the work of Horn [1986] for combining shading with contour, the work of Waxman et al. [Waxman et al., 1986] for combining stereo and motion, the work of Grimson [Grimson, 1984] for combining shading and stereo, the work of Richards and Huang and Blostein for stereo and motion [Richards, 1985; Huang and Blostein, 1985], and the work of Milenkovich and Kanade [Milenkovich et al., 1985]. So the need for such an approach has already been realized by some and the goal of this work is to contribute to a better understanding of this approach and that it will generate more related research.

The basic structure of the paper is depicted in the following diagram (Figure 3.2). In the ellipses (top) are the different image cues (we take the liberty to call stereo or motion a cue). It is obvious that by cue we mean a source of information, either coming from the image(s) or from the par-

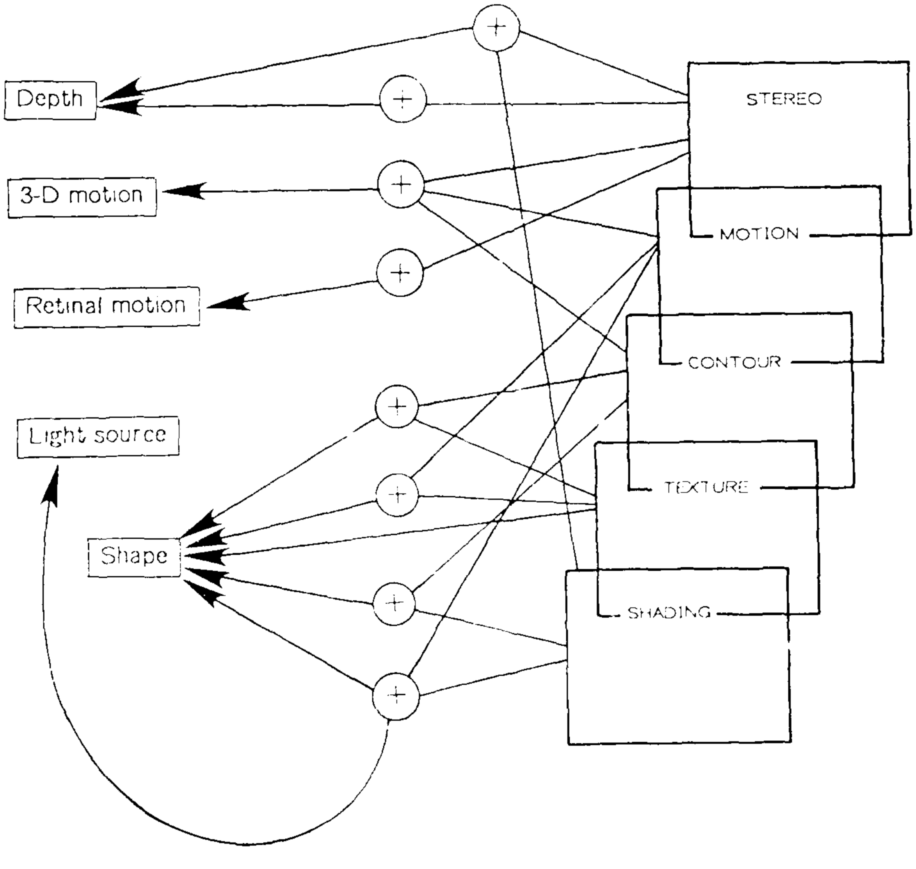


Figure 3.1 Proposed status

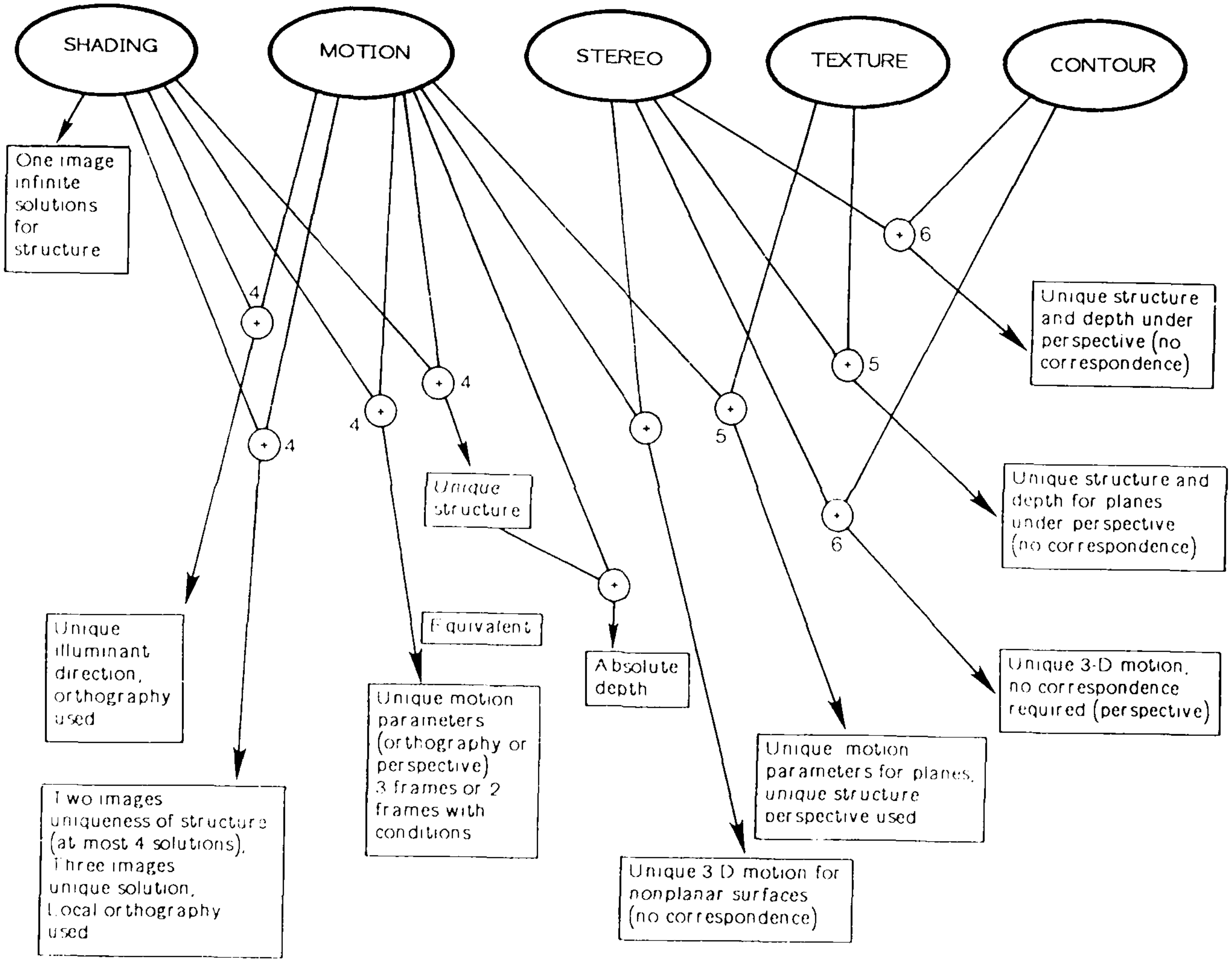

SHADING
MOTION
STEREO
TEXTURE
CONTOUR
One image infinite solutions for structure
4
4
4
4
5
5
6
6
Unique structure and depth under perspective (no correspondence)
Unique structure and depth for planes under perspective (no correspondence)
Unique 3-D motion, no correspondence required (perspective)
Unique motion parameters for planes, unique structure perspective used
Unique 3 D motion for nonplanar surfaces (no correspondence)
Unique structure
Absolute depth
Equivalent
Unique illuminant direction, orthography used
Unique motion parameters (orthography or perspective) 3 frames or 2 frames with conditions
Two images uniqueness of structure (at most 4 solutions), Three images unique solution, Local orthography used

Figure 3.2: Basic structure of the paper.

ticular setup or condition of the visual system (stereo-motion). In the squares are the results we obtain (in terms of propositions) when we combine information from two different cues. Two or more different cues are combined with arcs which lead to small circles containing a plus. Then, a different arc from the plus leads to a square containing the result from this combination. The numbers at a plus or an arc indicate that the theory for this particular computation can be found in the corresponding section. There are more combinations that one can make and the corresponding results may be found in [Aloimonos, 1986]. Also in the table there are some results that are not stated in the paper but they are easily inferrable.

3.4. Mathematical Preliminaries and What is to Come

It is very important to understand how the images are formed, because this is a prerequisite for being able to extract information from images. There are basically two questions about image formation:

a) What determines where the image of some point will appear?

b) What determines how bright the image of some surface will be?

Agreeing that *it is very important* to know how an image is formed in order to analyze it, we have to study two things: First, we need to find the geometric correspondence between points in the scene and points in the image, and second, we must find out what determines the brightness at a particular point in the image. Assuming that the reader is familiar with the concepts of perspective and orthographic projection as well as the meaning and representations of shape, depth, retinal and 3-D motion, reflectance models, and the like, we describe them in Appendix A.

Once again, this paper does not try to present a unified theory for the computation of the intrinsic images. Much more research is required for that, and the last section sheds some light on this issue. Instead, it proves that if several cues are combined and if the right (natural) assumptions are employed, then we can obtain visual computations which uniquely and robustly compute intrinsic images. We address the problems of shape from shading, shape from texture, and shape and 3-D motion from contour. The first three problems are *ill-posed while the last one has shown to be very difficult* in a practical situation (unstable) [Tsai and Huang, 1984]. Section 4 shows that if shading is combined with motion, then theoretically a unique solution is attainable. Section 5 shows that *if texture is combined with motion then again a unique* solution is guaranteed. Section 6 analyzes the problem of shape from contour and proves that if the information from the contour is combined with stereo, then uniqueness is guaranteed. Also, it is shown how to recover the 3-D motion of a planar contour without any correspondence, in the discrete case. Later sections analyze some algorithms from *the rest of the cues and the paper concludes with a summary* and some of our future work.

4. SHAPE FROM SHADING AND MOTION

In this section we prove that if we combine shading with motion, then we can uniquely compute the direction of the light source and the shape of the object in view. In particular:

(1) We develop a constraint between retinal motion displacements, *local shape and the direction of the light source*. It is worth noting that this constraint does not involve the albedo of the imaged surface. This constraint is of importance on its own, and it can be used in related research in computer or human vision.

(2) We develop a constraint between retinal displacements and local shape. Again, this constraint is important on its own, and it is the heart of the algorithm presented later in this section.

(3) We present algorithms for the unique computation of the lighting direction and the shape of the object in view.

(4) And we present some experimental results that test the theory.

The basic assumption in this section is that the retinal motion is computed everywhere in the image, in the case of a moving observer and a stationary scene, or a stationary observer and a moving object. If several objects are moving in the scene, then a segmentation is required first, i.e. the algorithms developed here can be applied to one rigidly moving object.

Shading is important for the estimation of three-dimensional shape from two dimensional images, for instance for distinguishing between the smooth occluding contour generated by the edge of a sphere and the sharp occluding contour generated by the edge of a disc. In order to successfully use shading, one must know the illuminant direction l. This is because variations in image intensity (shading) are caused by changes in surface orientation relative to the illuminant. This section reviews previous approaches to the solution of the determination of the illuminant direction problem and presents a new method for the unambiguous determination of the single lighting direction, and from that of the shape information. In particular, this part of the paper shows that if we combine information from shading and motion, then we can uniquely compute shape and the illuminant direction. Finally, in this section we are making the assumption of orthographic projection, since reflectance equation models are not known *up to this point under perspective projection.*[2]

4.1. Prerequisites

The ability to obtain three dimensional shape from two dimensional intensity images is an important part of vision. The human visual system in particular is able to use shading cues to infer changes in surface orientation fairly accurately, with or without the aid of texture of surface markings. An example in which shading information is important is the change in luminance that distinguishes a smooth occluding contour (such as that generated by the edge of a sphere) from a sharp occluding contour (such as that generated by the edge of a disc).

The direction of illumination is required to be known in order to obtain accurate three-dimensional surface shape from two dimensional shading because changes in image *intensity* are primarily a function of changes in surface orientation relative to the illuminant. For example, small changes in surface orientation parallel to the illuminant direction can cause large changes in image intensity, whereas large changes in surface orientation that occur in a direction perpendicular to the

[2] But recent research shows that the model might be the same for perspective [Shafer, S., private communication, 1987].

direction of illumination will not change image intensity at all. So, the illuminant direction must be known before one can determine what a particular change in image intensity implies about changes in surface orientation. In the section, we develop a computational theory for the determination of the illuminant direction, and the shape of the object in view, from two images of a moving object (or from two images taken by a moving observer). Before we proceed, we should discuss a little about image formation, even though this was discussed in Section 2, in general terms.

4.2. Process of Image Formation

In order to be able to make quantitative statements about the world and the image and specifically to estimate the illuminant direction, we must use a mathematical model for the image formation. A great deal of work has been done in this area [Horn et al., 1975, 1979] and many models have been developed. For the purposes of the section, we use the following simple and universally accepted model (see Figure 4.1). Assuming orthographic projection, if **n** is the surface normal at a point on th• imaged surface, **l** is the illuminant direction and f is the flux emitted towards the surface and we assume a Lambertian reflectance function for the surface [Horn et al., 1975, 1979], the image intensity is given by

$$I = \rho f(\mathbf{n} \cdot \mathbf{l})$$

where ρ is the albedo of the surface, a constant depending on the surface. For more on the subject, see Appendix A.

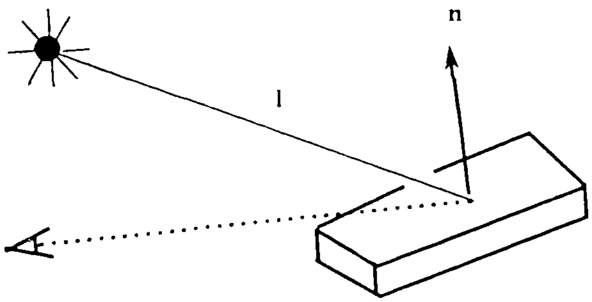


Figure 4.1: Process of image formation.

4.3.1. Motivation and Previous Work

Despite the fact that the problem of determining the illuminant direction is important for computer vision (shape from shading), not too much work has been done towards its solution.

We stress here the fact that the problem of the determination of the illuminant direction is important. Most of the work in shape from shading [Horn, 1975; Strat, 1979; Ikeuchi et al., 1981] assumes that the albedo of the surface in view and the illuminant direction are known *a priori*; in other words, this work assumes that the reflectance map specifies how the brightness of a surface parch depends on its orientation, under given circumstances. It therefore encodes information about the reflecting properties of the surface and information about the distribution and intensity of the light sources. In fact, the reflectance map can be computed from the bidirectional reflectance-distribution function and the light source arrangements, as shown by [Horn and Sjoberg, 1979].

When encountering a new scene, we usually do not have the information required to determine the reflectance map. Yet, without this information, we are unable to formulate the shape from shading problem, much less solve it.

The dilemma may be resolved if a calibration object of known shape appears in the scene, since the reflectance map can be computed from its image; but what happens when we are not that fortunate? It is evident from the above discussion that at least the knowledge of the illuminant direction is required. The only work done on the illuminant direction determination is due to [Pentland, 1982, 1984; Brooks, 1985; and Brown et al., 1983]. Pentland's method is based on the assumption that surface orientation, when considered as a random variable over all possible scenes, is isotropically distributed. A consequence of this assumption is that the change of surface normals is also isotropically distributed. Pentland's method, which uses the same model of image formation that we do, is valid for some objects. Under his assumptions, Pentland solves the problem uniquely, but his assumptions are very restrictive.

On the other hand, Brooks and Horn [1985] presented a method in the general framework of ill-posed problems and regularization in early vision. Their theory proposes to solve the shape form shading problem and at the same time to compute the illuminant direction, by minimization of an appropriate funtional. They did not present any uniqueness or convergence proofs of their iterative methods, but their experimental results for synthetic images were reasonable.

Finally, [Brown et al., 1983] presented a method that, based on Lambertian reflectance and a Hough transform technique, attempted the recovery of the direction of the light source.

In the sequel, we prove that the illuminant direction cannot be recovered from only one intensity image of a Lambertian surface. After this, we will prove that two intensity images (moving object or moving observer), with the correspondence between them established, can uniquely recover the illuminant direction, and from that the shape.

4.4. A Uniqueness Result

Here we prove the following theorem.

THEOREM 1: *Given an image (i.e. an intensity function $I(x,y)$), there are an infinite number of surfaces and an infinite number of positions of the light source that will produce the same image under the process of image formation described in Section 4.2.*

Proof: Suppose that for a shape $\mathbf{n}_1(x,y)$, $(x,y) \in \Omega$ (Ω is the domain where the image function is defined) and a light source position $\mathbf{s}_1$ we have $I(x,y) \quad \rho\, \mathbf{n}_1(x,y)\ \mathbf{s}_1$, where ρ is the albedo of the surface in view (considered constant everywhere). Define a shape $\mathbf{n}_2(x,y)$ over Ω and a light source position $\mathbf{s}_2$ as follows: $\mathbf{n}_2(x,y) \quad 2\mathbf{m}(\mathbf{n}_1(x,y) \cdot \mathbf{m}) \quad \mathbf{n}_1(x,y)$, $\mathbf{s}_2 \quad 2\mathbf{m}(\mathbf{s}_1 \cdot \mathbf{m}) \quad \mathbf{s}_1$ for any vector $\mathbf{m}$ with $\|\mathbf{m}\| \quad 1$

Then, considering a surface with the same albedo as before and with shape $\mathbf{n}_2(x,y)$ and illuminated from a point source in the direction $\mathbf{s}_2$, we have

$$\rho\, \mathbf{n}_2(x,y)\cdot \mathbf{s}_2 = \rho(2\mathbf{m}(\mathbf{n}_2(x,y)\cdot \mathbf{m}) - \mathbf{n}_1(x,y))\cdot(2\mathbf{m}(\mathbf{s}_1\, \mathbf{m}) - \mathbf{s}_1)$$

$$= \rho\, [4(\mathbf{m}\cdot\mathbf{m})(\mathbf{n}_1(x,y)\cdot\mathbf{m})(\mathbf{s}_1\cdot\mathbf{m}) - 2(\mathbf{n}_1(x,y)\cdot\mathbf{m})$$

$$\cdot(\mathbf{s}_1 - \mathbf{m}) - 2(\mathbf{s}_1\cdot\mathbf{m}) + \mathbf{n}_1(x,y)\cdot\mathbf{s}_1]$$

$$= \rho\, \mathbf{n}_1(x,y)\cdot\mathbf{s}_1$$

So, $I(x,y) = \rho\, \mathbf{n}_2(x,y)\cdot\mathbf{s}_2$.

This means that the image $I(x,y)$ could be due to an infinite number of surfaces illuminated from one of an infinite number of light sources, since the vector $\mathbf{m}$ can be arbitrary. (q.e.d.)

The importance of the above theorem is that no correct and robust method can exist that will find the illuminant direction from one intensity image of a Lambertian surface illuminated from a point source.

We now move to the main part of this section, that is a theorem that states that given two images of a moving object (or two images of the same object taken by a moving observer), with the correspondence between the two images established, the position of the light source can be uniquely determined. But before that, we need some technical prerequisites that are presented in the following section.

4.5. Technical Prerequisites

In this section we develop two technical results, one concerning the relationship between shape, intensity, displacements and the lighting direction and the other concerning the parameters of a small motion (small rotation) with the shape. We begin with the following theorem.

THEOREM 2: *Suppose that two views (rigid motion) of the same (Lambertian) surface (locally planar) are given and let I_1 and I_2 be the two intensity functions. Suppose also that the displacement vector field $(u(x,y),\ v(x,y))$, $(x,y)\in\Omega$ is known, where Ω is the domain of the image, i.e. a point (x,y) in the first image will move 'o the point $(x + u(x,y),\ y + v(x,y))$ in the second image. If the lighting direction is $\mathbf{l} = (l_1, l_2, l_3)$ and the gradient of a surface point whose image is the point (x,y) is (p,q), the following relation holds:*

$$p^2[(l_2\Delta^y u - l_1(1 + \Delta^y v))^2 - r^2 l_1^2] +$$

$$2pq[(l_1\Delta^x v - l_2(1 + \Delta^x u))(l_1(1 + \Delta^y u) - l_2\Delta^y u) - 2r^2 l_1 l_2] +$$

$$q^2[(l_1\Delta^x v - l_2(1 + \Delta^x u))^2 - r^2 l_2^2 -$$

$$2pl_1 l_3 r(r - ((1 + \Delta^x u)(1 + \Delta^y v) - \Delta^x u\, \Delta^x v)) -$$

$$2ql_2 l_3 r(r - ((1 + \Delta^x u)(1 + \Delta^y v) - \Delta^y u\, \Delta^x v)) -$$

$$((1 + \Delta^x u)(1 + \Delta^y v) - \Delta^y u\, \Delta^x v)^2 +$$

$$l_1^2((\Delta^x v)^2 + (1 + \Delta^y v)^2) + l_2^2((\Delta^y u)^2 + (1 + \Delta^x u)^2) +$$

$$+ l_1 l_2((1 + \Delta^x u)\Delta^x v + \Delta^y u(1 + \Delta^y v)) -$$

$$- r^2 l_3^2 - 2rl_3^2((1 + \Delta^x u)(1 + \Delta^y v) - \Delta^y u\, \Delta^x v) = 0 \qquad (4.1)$$

where

$$r - \frac{I2(x + u(x,y),\, y + u(x,y))}{I1(x,y)}$$

and

$$\Delta^x u = u(x + 1,\, y) - u(x,y)$$

$$\Delta^y u = u(x,y + 1) - u(x,y)$$

$$\Delta^x v = v(x + 1,\, y) - v(x,y)$$

$$\Delta^y v = v(x,y + 1) - v(x,y)$$

It is clear that the above equation (4.1) is local, i.e. it involves the gradient at a point (x,y), the displacements around the point (x,y) along with the global direction of lighting. The above constraint is a conic in p,q of the form

$$Ap^2 + Bq^2 + Cpq + Dp + Eq + F = 0,$$

*where the coefficients depend on local displacements and the light source direction. From now on we will refer to equation (4.1) as the **lighting constraint**.*

Proof: To exploit the rigid motion assumption, we represent the surface normal by two vectors and note that their length, angular separation and hence their dot and cross product are preserved by rigid motion. Consider the surface S, a point A on S, the vector $\mathbf{n} = p\mathbf{i} + q\mathbf{j} + \mathbf{k}$ perpendicular to S at the point A, and the plane Π that is tangent to the surface S at the point A (see Figure 4.2). The vectors $\mathbf{a} = (1,0,-p)$ and $\mathbf{b} = (0,1,-q)$ lie in Π and

$$\mathbf{a} \times \mathbf{b} = p\mathbf{i} + q\mathbf{j} + \mathbf{k} = \mathbf{n}$$

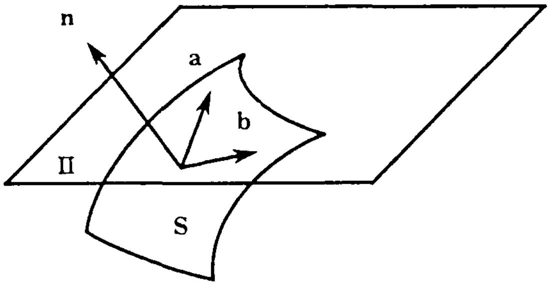


Figure 4.2: Shape vectors.

We shall use vectors a and b as the shape (surface normal) representation (Figure 4.3). We use the following traditional camera model. Let O be the position of the nodal point of the eye, let $OXYZ$ be a coordinate system that is fixed with respect to the eye, and let OZ be the line of sight. Finally, let the image plane be perpendicular to the Z-axis at the point (0,0,1).

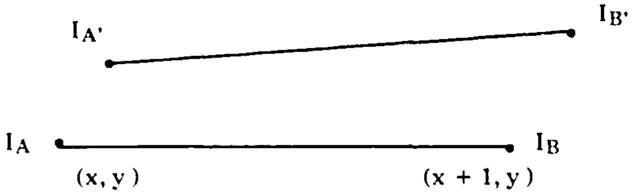


Figure 4.3: Displacement vectors.

Consider a point $A(x,y)$ on the surface S at time t whose shape vectors are $\mathbf{a} = \mathrm{vec}(AB) = \mathbf{i} - p\mathbf{k}$ and $\mathbf{b} = \mathrm{vec}(AC) = \mathbf{j} - q\mathbf{k}$ ($\mathrm{vec}(AB)$ means the vector from the point A to the point B). Since the projection of $\mathbf{a} = \mathrm{vec}(AB)$ on the image plane is $\mathbf{i}$, we conclude that if $I_A = (x,y)$ is the projection of A then the projection of B will be $I_B = (x+1, y)$. Similarly, the projection of C will be the point $I_C = (x, y+1)$ on the image plane. Consider now the object at the next frame where the point A will become A', with shape vectors $\mathbf{a}' = A'B'$ and $\mathbf{b}' = A'C'$.

Let $I_{A'}$ be the projection of A'. $I_{A'}$ is the position to which I_Z moves, which can be determined from the displacements. Thus, $I_{A'} = (x + u(x,y),\ y + v(x,y))$. Similarly, if $I_{B'}$ and $I_{C'}$ are the projections of B' and C', then $I_{B'}$ is the position to which I_B will move. Of course, the displacement at I_B is due to the motion of the surface point that has the same x,y coordinates as B (orthography), but because of the assumed local planarity, this point is the same as B. (The planarity constraint of course fails at boundaries.) So, we have $I_{B'} = (x + 1 + u(x+1, y),\ y + v(x+1, y))$, and for the same reasons $I_{C'} = (x + u(x, y+1),\ y + 1 + v(x, y+1))$. The projection of $\mathbf{a}' = \mathrm{vec}(A'B')$ on the image plane is thus

$$I_{A'}I_{B'} = [1 + u(x+1, y) - u(x,y)]\mathbf{i} + [v(x+1, y) - v(x,y)]\mathbf{j}$$

But according to our hypotheses, the above relation can be written

$$I_{A'}I_{B'} = (1 + \Delta^x u)\mathbf{i} + \Delta^x v_j$$

Similarly, we get

$$I_{A'}I_{C'} = \Delta^y u\,\mathbf{i} + (1 + \Delta^y v)\mathbf{j}$$

The above two equations give us the expressions for $I_{A'}I_{B'}$ and $I_{A'}I_{C'}$ which are the projections of the shape vectors $\mathbf{a}'$ and $\mathbf{b}'$ respectively. But then

$$\mathbf{a}' = (1 + \Delta^x u)\mathbf{i} + \Delta^x v\mathbf{j} + \lambda\mathbf{k} \text{ and}$$

$\mathbf{b}' = \Delta^y u\,\mathbf{i} + (1 + \Delta^y v)\mathbf{j} + \mu\mathbf{k}$ where λ, μ are to be determined.

But since rigid body motion preserves the vector length, we have

$$||\mathbf{a}'||^2 = ||\mathbf{a}||^2 \text{ or}$$

$$\lambda = \pm(p2 - \Delta^x u - \Delta^x v - 2^*\Delta^x u)^{1/2}$$

Similarly, we get

$$\mu = \pm(q2 - \Delta^y u - \Delta^y v - 2^*\Delta^y u)^{1/2}$$

Assuming that neither region is in shadow, we have

$$I_1(x,y) = \rho_1 \mathbf{n_A} \mathbf{l} \tag{4.2}$$

and

$$I_2(x + u(x,y),\ y + v(x,y)) = \rho_2 \mathbf{n_{A'}} \mathbf{l} \tag{4.3}$$

Equation (4.2) gives the intensity of the point $A(x,y)$ in the first frame. Note that $\mathbf{n_A}$ is the surface normal at the point A (in the first view). Equation (4.3) gives the intensity at the point $A'(x + u(x,y),\ y + v(x,y))$ in the second frame. Note that $\mathbf{n_{A'}}$ is the surface normal at the new position of the point A.

Dividing equations (4.2) and (4.3) and setting $I2(A')/I1(A) = r$ and taking into account that $\rho_1 = \rho_2$ (surface markings do not change), we get

$$r\,\mathbf{n_A}\cdot\mathbf{l} = \mathbf{n_{A'}}\cdot\mathbf{l} \tag{4.4}$$

But

$$\mathbf{n_A} = \frac{\mathbf{a} \times \mathbf{b}}{||\mathbf{a} \times \mathbf{b}||} \tag{4.5}$$

and from the rigidity of the motion it follows that

$$\mathbf{n_{A'}} = \pm\frac{\mathbf{a}' \times \mathbf{b}'}{||\mathbf{a}' \times \mathbf{b}'||} \tag{4.6}$$

where the sign is chosen such that $\mathbf{n_{A'}}\cdot\mathbf{k} > 0$. But since $||\mathbf{a}'|| = ||\mathbf{a}||$, $||\mathbf{b}'|| = ||\mathbf{b}||$ and $\mathbf{a}\cdot\mathbf{b} = \mathbf{a}'\cdot\mathbf{b}'$, it follows that

$$||\mathbf{a} \times \mathbf{b}|| = ||\mathbf{a}' \times \mathbf{b}'|| \tag{4.7}$$

Using equation (4.7), equation (4.4) becomes

$$r[\mathbf{a},\mathbf{b},\mathbf{l}] = \pm[\mathbf{a}',\mathbf{b}',\mathbf{l}'] \tag{4.8}$$

where the sign chose is the sign of $[\mathbf{a}', \mathbf{b}', \mathbf{k}]$ and $[\,,\,,\,]$ is the triple scalar product of vectors. But $[\mathbf{a}', \mathbf{b}', \mathbf{k}] = (1 + \Delta^x u)(1 + \Delta^y v) - \Delta^y u\,\Delta^x v$. It is obvious that $[\mathbf{a}, \mathbf{b}, \mathbf{k}] > 0$; if $[\mathbf{a}', \mathbf{b}', \mathbf{k}] < 0$, then we don't have a valid motion, because $[\mathbf{a}', \mathbf{b}', \mathbf{k}] < 0$ means that we have reversed orientation so that the texture in the image is viewed as if seen in a mirror. So, $[\mathbf{a}', \mathbf{b}', \mathbf{k}] > 0$, and substituting in equation (4.8) the values of $\mathbf{a}$, $\mathbf{b}$, $\mathbf{a}'$, $\mathbf{b}'$ after algebraic manipulations and using the fact that

$$\lambda^2 = p^2 + 1 - (1 + \Delta^x u)^2 - (\Delta^x v)^2$$

$$\mu^2 = q^2 + 1 - (1 + \Delta^y v)^2 - (\Delta^y u)^2$$

$$\lambda\mu = pq - (1 + \Delta^x u)\Delta^y u - (1 + \Delta^y v)(\Delta^x v)$$

we get equation (4.1).

It is obvious that equation (4.1) involves the lighting direction, but it also involves the shape (p,q). We would like to find the direction $\mathbf{l}$ without knowing the shape (p,q), otherwise the problem is of no importance. Theorem 2 is very important, in the sense that is has established a constraint between lighting direction, shape and displacement vectors.

We now proceed with the following theorem.

THEOREM 3: *Suppose that the surface S (locally planar) is moving with a rigid motion, and the camera model is the one described in the previous theorem. Let the gradient of the surface (with respect to the first frame) be $(p(x,y),\ q(x,y))$ and the displacement vector field be $(u(x,y),\ v(x,y))$. It is known that this motion can be considered as a translation $(dx,\ dy,\ dz)$ plus a rotation of an angle θ about an axis (n_1, n_2, n_3) passing through the origin $(n_1^2 + n_2^2 + n_3^2 = 1)$. If the rotation angle θ is small, then the following relation holds:*

(a) *The displacement vector field $(u(x,y),\ v(x,y))$ is given by*

$$u(x,y) = dx + Bz(x,y) - Cy$$

$$v(x,y) = dy + Cx - Az(x,y).$$

where $A = n_1\theta$, $B = n_2\theta$, $C = n_3\theta$ and $z(x,y)$ the depth of the surface point whose projection is the image point (x,y).

(b)

$$p(x,y) = \frac{1}{B}\Delta^x u$$

$$q(x,y) = \frac{1}{A}\Delta^y v$$

$$\frac{A}{B} = \frac{\Delta^y v}{\Delta^x u} \cdot \frac{\frac{\Delta^x v - \Delta^y u + \sqrt{(\Delta^y u + \Delta^x v)^2 - 4 * \Delta^x u * \Delta^y v}}{2} - \Delta^x v}{\Delta^y v}$$

Proof:

(a) Trivial.

(b) Using the two equations in (a) and the assumption about local planarity, we get

$$\Delta^x u = Bp$$

$$\Delta^x v = C - Ap$$

$$\Delta^y u = Bq - A$$

$$\Delta^y v = -Aq$$

where the p and q are considered at the point (x,y). From the above equations we get

$$p(x,y) = \frac{1}{B}\Delta^x u$$

$$q(x,y) = \frac{1}{A}\Delta^y v$$

$$\frac{A}{B} = \frac{\Delta^y v}{\Delta^x u} \cdot \frac{\frac{\Delta^x v - \Delta^y u + \sqrt{(\Delta^y u + \Delta^x v)^2 - 4 * \Delta^x u * \Delta^y v}}{2} - \Delta^x v}{\Delta^y v}$$

(q.e.d.)

4.6. Development of the Lighting Direction Constraint

In this section we develop the constraint that will be used as the heart of the algorithm that will solve the problem of the determination of the illuminant direction. If we let $1/A = \alpha$, $1/B = \beta$, $B/A = K$ and use part (b) of Theorem 3, to substitute in equation (4.1) for p and q, we get the following equation.

$$(\Delta^x u)^2\beta^2[(l_2\Delta^y u - l_1(1 + \Delta^y v))^2 - r^2 l_1^2] +$$

$$+2\Delta^x u \Delta^y v K \beta^2[(l_1\Delta^x u - l_2)(1+\Delta^x u))(l_1(1+\Delta^y v) - l_2\Delta^y u) - 2$$

$$r^2 l_1 l_2] + (\Delta^y v)^2 K^2 \beta^2[(l_1\Delta^x v - l_2(1 + \Delta^x u))^2 - r^2 l_2^2] -$$

$$-2(\Delta^x u)\beta l_1 l_3 r (r - (1 + \Delta^x u + \Delta^y v + \Delta^x u \Delta^y v - \Delta^y u \Delta^x v)) -$$

$$-2(\Delta^y v) K \beta l_2 l_3 r (r - (1 + \Delta^x u + \Delta^y v + \Delta^x u \Delta^y v - \Delta^y u \Delta^x v)) -$$

$$-(1 + \Delta^x u \Delta^y v + \Delta^x u + \Delta^y v - \Delta^y u \Delta^x v)) +$$

$$+ l_1^2((\Delta^x v)^2 + (1 + \Delta^y u)^2) + l_2^2((\Delta^x u)^2 + (1 + \Delta^x u)^2) +$$

$$+ l_2 l_1((1 + \Delta^x u)\Delta^x v + \Delta^y u(1 + \Delta^y v)) - r^2 l_3^2 +$$

$$+ 2rl_3^2(1 + \Delta^x u + \Delta^y v + \Delta^x u \Delta^y v - \Delta^y u \Delta^x v) = 0 \qquad (4.9)$$

The above equation is a polynomial in l_1, l_2, l_3, β.

Considering equation (4.9) in four points we get a polynomial system of four equations in four unknowns. A simple but tedious calculation of the Jacobian of this system gives us the fact that the Jacobian has rank four (except for the degenerate cases whose set has measure zero). This means (inverse function theorem) that the function defined by the equations of the system is locally an isomorphism, which means that its zeros are isolated. But, from Whitney's theorem, the set of the zeros of this algebraic system is an algebraic set and it has finitely many components.

The conclusion of this is that the solutions of the system are finite (uniqueness). If we now consider equation (4.9) in five points, then we get a system of five equations in four unknowns which, barring degeneracy, will have at most one solution.

It is clear from the above discussion that two intensity images of a Lambertian surface, with the correspondence between them established, gives the lighting direction uniquely. In the next section, we present a practical way to recover the lighting direction based on the constraint developed in this section.

4.7. The Algorithm for Finding Illuminant Direction

First of all we choose the Gaussian sphere formalism (azimuth-elevation) to represent the vector that denotes the lighting direction. More specifically, we set

$$l_1 = \cos\phi\cos\theta$$

$$l_2 = \sin\theta$$

$$l_3 = \cos\phi\cos\theta$$

where θ and ϕ are the azimuth and elevation (see Figure 4.4). Now we consider equation (4.9) in n points in the images, and we get n equations $eq1$, $eq2$, . . . , eqn in the three unknowns β, θ, ϕ. Then, the following algorithm solves the problem:

```
for all $\theta$
    for all $\phi$
        {
            get $n$ quadratic equations in $\beta$
            Check if they have a common solution
            If yes, output $\theta$ $\phi$.
        }
```

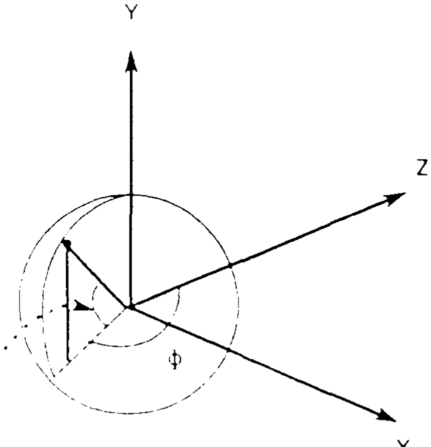


Figure 4.4: Gaussian sphere.

4.8. Applying the Algorithm to Natural Images

When one is experimenting with natural images, it is sometimes difficult to compute the displacement field for every point in the image. In that case, one can compute the parameters of the correspondence of small regions. In other words, if we have a small planar region S_1 in the first image that corresponds to a small region S_2 in the second image, then the parameters of an affine transformation $f(x,y) - (ax + by + c,\ dx + cy + f)$ between the two patterns (see Figure 4.5) can be computed using a variation of a least-squares method introduced by Lucas and Kanade that is described in Webb [1983]. In that case, the essential constraint (equation (4.9)) has a similar form and the whole analysis proceeds as previously.

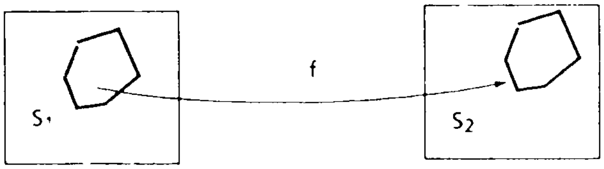


Figure 4.5: Corresponding regions.

4.9. Implementation and Experiments

We have implemented the above mentioned algorithm, and it works successfully for synthetic images. Figure 4.6 represents the displacement vector field that was obtained from the motion (rotational with $\omega_x \quad -1$, $\omega_y \quad 2$, $\omega_z \quad 3$) of a sphere. Figure 4.7 represents the image of the sphere before the motion. The surface of the sphere is supposed to be Lambertian, the albedo $\rho \quad 1$ and the lighting direction with gradient $(ps,qs) \quad (0.7,\ 0.3)$, i.e. to the right and a little above

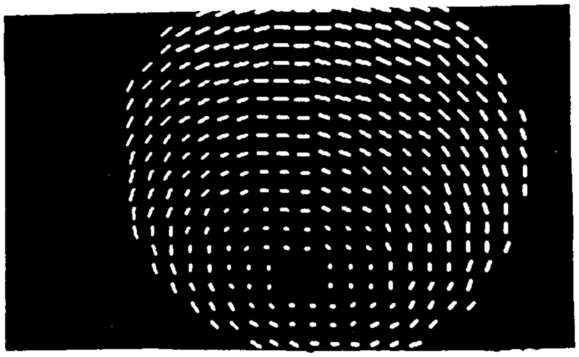


Figure 4.6: Displacements field.

the horizon. The computed lighting direction was (0.65, 0.33). The observed inaccuracy is due to discretization effects. In our synthetic experiment, we did not compute the displacements; instead, we calculated them since we knew the motion and the exact position of the sphere. If the Lambertian reflectance model is not adequate for capturing natural images (which it is not, of course), there probably exists a model (not discovered yet) that captures natural shading. This model, should of course depend on the shape and the lighting direction. The approach that we took here can be taken with any other model of reflectance, and it is one of our future goals to apply the method in natural images and employ general reflectance models, that consider the illumination from the sun and the sky.

The next section discusses the problem of determining shape from shading and motion in a unique way. Again the findings of the next section cannot be applied at this point to

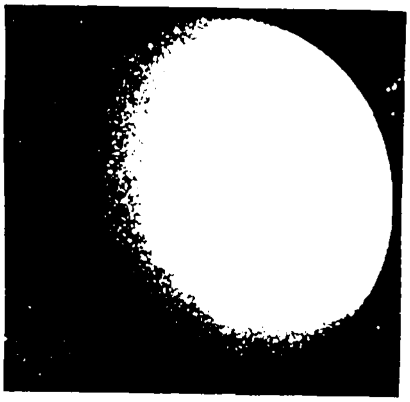


Figure 4.7. Intensity image for a sphere.

natural images, for the reasons that we mentioned above. The treatment again is of theoretical value, and the method can be applied to natural imagery, if better reflectance models (i.e. models that capture the reality) were known, and the computation of retinal motion in natural images were feasible.

4.10. Computing Shape from Shading and Motion

In this section we discuss the problem of determining shape from shading and motion. Before we proceed we need some technical prerequisites, that are introduced in the next sections.

4.11. The Constraint Between Shape and Displacements

THEOREM 4: *With the assumptions and notation of Theorem 2, the gradient (p,q) of a surface point whose image is the point (x,y), with displacement vector (u,v), satisfies the constraint*

$$Ap^2 + Bq^2 \quad Cpq + D \quad 0, \text{ with}$$

$$A - (\Delta^y u(x,y))^2 + (\Delta^y v(x,y))^2 + 2\Delta^y v(x,y)$$

$$B \quad (\Delta^x u(x,y))2 + (\Delta^x v(x,y))2 + 2\Delta^x u(x,y)$$

$$C \quad \Delta^y u(x,y) + \Delta^y u(x,y)\Delta^x u(x,y) + \Delta^x v(x,y)$$
$$+ \Delta^x v(x,y)\Delta^y v(x,y)$$

$$D \quad C^2 \quad AB,$$

where the coefficients $\Delta^x u$, $\Delta^y u$, $\Delta^x v$, $\Delta^y v$ are defined in Theorem 2.

Proof: From the proof of Theorem 2, because of the rigid motion assumption, we have the preservation of the dot product. So,

$$\mathbf{a}\cdot\mathbf{b} \quad \mathbf{a}'\cdot\mathbf{b}'$$

or

$$Ap^2 + Bq^2 \quad 2Cpq + D \quad 0 \tag{4.10}$$

where (p,q) is the gradient at the point (x,y) and

$$A \quad (\Delta^y u(x,y))^2 + (\Delta^y v(x,y))^2 + 2\Delta^y v(x,y)$$

$$B \quad (\Delta^x u(x,y))2 + (\Delta^x v(x,y))2 + 2\Delta^x u(x,y)$$

$$C = \Delta^y u(x,y) + \Delta^y u(x,y)\Delta^x u(x,y) + \Delta^x v(x,y)$$
$$+ \Delta^x v(x,y)\Delta^y v(x,y)$$

$$D = C^2 - AB$$

Equation (4.10) gives the constraint between displacements and shape and represents a conic section in p-q space. This conic section is a hyperbola or parabola depending on the values of the coefficients A, B, C. The constraint (4.10) is a constraint between shape and displacements. Constraint (4.1) is a constraint between shape, displacements and the lighting direction. For the purposes of the rest of this section, and to avoid confusion, we will refer to constraint (4.10) as the *shape-motion* constraint.

4.12. How to Utilize the Constraints

Here we show how to utilize the constraints to recover the three-dimensional shape of the object in view, using shading and motion. It is assumed that the lighting direction has already been computed with the algorithm described in Section 4.7. Up to now we have developed two constraints on shape, that also involve retinal motion displacements and the lighting direction. The lighting constraint is a conic on p,q with coefficients that depend on intensities (relative), displacements and lighting direction. The shape-motion constraint is again a conic on p and q, with coefficients that depend on the displacement vectors. Finally, the image irradiance equation

$$I = \rho f(\mathbf{n} \cdot \mathbf{l}) \qquad (4.11)$$

that determines the intensity $I(x,y)$ at a point (x,y) of the image, as a function of the shape $\mathbf{n}$ of the world point whose image is the point (x,y) and the lighting direction $\mathbf{l}$, is another constraint on p and q that is also a conic. This constraint we will call the *image-irradiance* constraint. The next subsections will describe algorithms for the unique computation of shape, under a variety of situations.

Figure 4.7.1 below gives a geometrical description of the constraints.

4.12.1. Computing shape when the albedo is known (numerical techniques)

When the albedo is known, then we have at our disposal three constraints at every image point for the computation of shape. The lighting constraint, let it be $F_1(p,q) = 0$, the shape-motion constraint, let it be $F_2(p,q) = 0$, and the image-irradiance constraint, let it be $F_3(p,q) = 0$. These are three equations, all of degree two, and their system, barring degeneracy, will have at most one solution. Several algebraic or geometrical techniques exist for solving a system of algebraic equations, each of degree two. Direct methods result in solving equations of a high degree, and so we prefer to use an iterative technique; and even though we do not have theoretical results about the convergence of the technique, in practice it has shown to converge very fast, to the right solution.

The function $E(p,q) = \lambda_1(F_1(p,q))^2 + \lambda_2(F_2(p,q))^2 + \lambda_3(F_3(p,q))^2$ should be minimized everywhere in the image, where λ_1, λ_2, λ_3 are constants with their sum equal to one. If

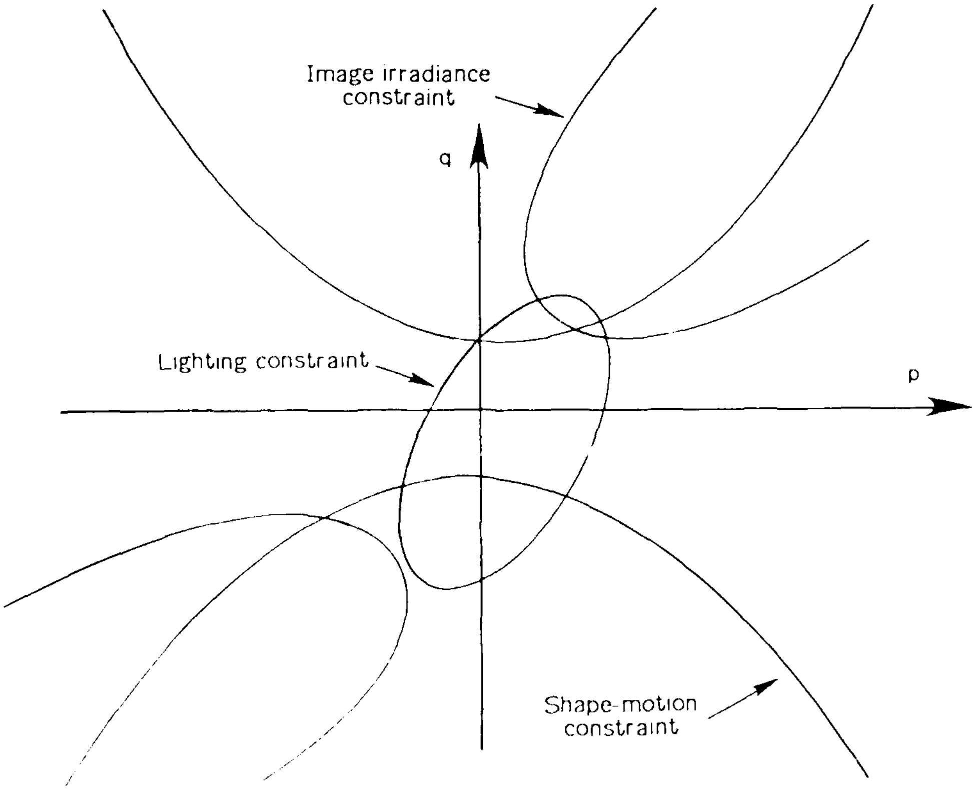


Figure 4.7.1: Pictorial description of the constraints.

we add one more term in the error function that accounts for smoothness and by setting the partial derivatives of $E(p,q)$ equal to zero and solving for p and q, we get equations of the following form: $p = G_1(p,q,p_{av})$, $q = G_2(p,q,q_{av})$, where G_1 and G_2 are polynomials in p,q,p_{av} and p,q,q_{av} respectively. These equations can be solved iteratively, provided that we have an approximate initial solution. If the values of p and q at the boundaries are known, then p,q are propagated throughout the image using a general smoothness criterion, by an algorithm similar to the Gauss-Seidel algorithm of Ikeuchi and Horn [Ikeuchi and Horn, 1981]. At this point we should emphasize that we do not depend on smoothness to achieve uniqueness in our methods. Smoothness is used to achieve an approximate initial solution.

4.12.2. Computing shape when the albedo is not known

a) **Iteratively**.

If the albedo is not known, then we cannot utilize the image-irradiance constraint, because it contains the albedo as a coefficient. We have to use the lighting constraint and the shape-motion constraint. An algorithm similar to the one in the above section can be easily obtained. At this point, the uniqueness of this problem has to be discussed. The lighting constraint and the shape-motion constraint are two conics in p and q. The Jacobian of the system that they form is non-zero. So, the function defined by the system is locally an isomorphism (from the inverse function theorem), which means that its zeros are isolated. But from Whitney's theorem, the set of zeros of this algebraic system is an algebraic set and it has finitely many components. The conclusion of this is that *the system has finitely many solutions*. In this case, the solutions might be restricted to a unique solution, if a local smoothness constraint is used. This, being in the paradigm of regularization theory, has been observed from experiments, and up to this point we do not have a formal proof.

b) **Directly**.

In a minimization scheme based on the Lagrange multiplier technique, the solution is obtained without propagating the boundary conditions. If $F_1(p,q)$ 0 is the lighting constraint and $F_2(p,q)$ 0 the shape-motion constraint, the error function $E(p,q)$ $(F_1(p,q))^2$ is to be minimized subject to the constraint $F_2(p,q) = 0$. The Lagrange multiplier scheme says that the p,q that minimize E are one of the solutions of the following system:

$$\nabla E \quad \lambda \nabla F_2,\ F_2 \quad 0 \text{ where } \lambda \text{ is the Lagrange multiplier.}$$

4.12.3. Implementation and experiments

Figures 4.8 and 4.9 represent exactly the same entities as Figure 4.6 and 4.7. From this input, our algorithm (4.12.1) recovered the shape shown in Figure 4.10. A local smoothing scheme has been used at the end of the program to smooth out the results. The error in the resulting shape is very low, basically due to discretization effects. If we introduce noise in the input, then the results get very much corrupted, if we don't apply a smoothing scheme, because of the locality of the method. If a local smoothness constraint is utilized, then the results are very good.

4.13. Conclusions and Future Directions

In this section we have presented a theory on how to compute in a unique way shape and the direction of the light

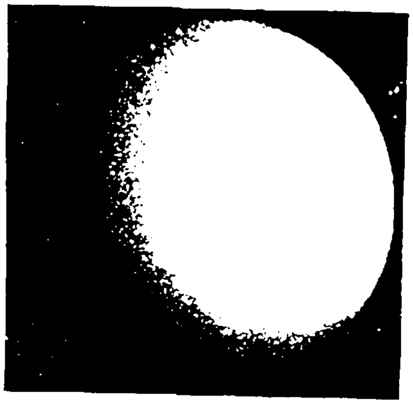


Figure 4.8: Intensity image for a sphere.

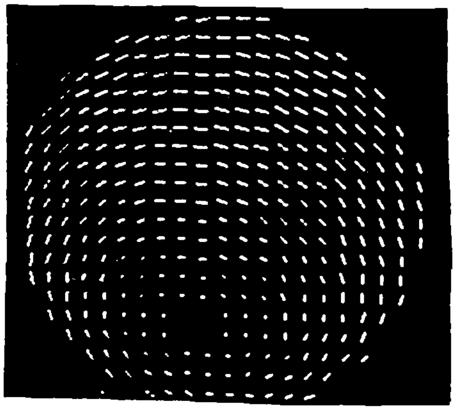


Figure 4.9: Displacements field.

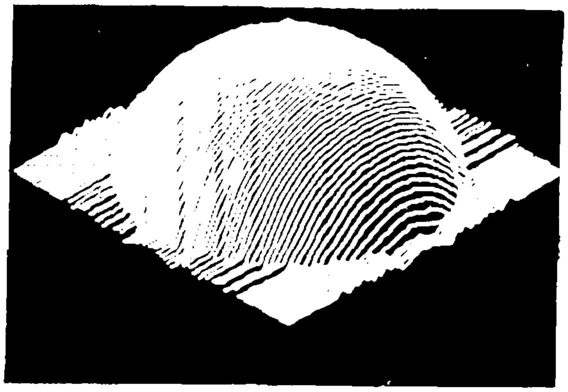


Figure 4.10: Reconstructed shape.

source, from shading and motion, under orthographic projection. Our input is the intensity of two images in a dynamic sequence, with the correspondence between the two frames established. We proved that in this case the light source direction and the shape of the object in view can be uniquely determined. Our results have theoretical value, since they demonstrate that if we combine information from different sources then we can obtain unique results for intrinsic images. In the past, there has been in this framework only the work of Grimson [Grimson, 1984], that combined shading with stereo with reasonable results. It is one of our future goals to extend this theory to capture a very general reflectance map [Brooks and Horn, 1985], that models the illumination due to the sun and the sky.

5. SHAPE FROM TEXTURE AND MOTION

In this section we study the problem of the perception of shape from texture and motion. We prove that a monocular observer that moves with a known motion can uniquely recover the gradient of the plane in view. In particular:

(1) Our theory does not require the finding of any correspondence.

(2) There are no assumptions about the texture.

(3) The computation of spatial derivatives of the image intensity function (an ill-posed problem) is not needed.

The problem of shape from texture has received a lot of attention in the past few years and some excellent research on the topic has been published [Stevens, 1979; Witkin, 1980; Davis et al., 1983; and Kanatani, 1984]. The problem is defined as "finding the orientation of a textured surface from a static monocular view of it." This problem is ill-posed in the sense that there exist infinitely many solutions. To restrict the space of solutions, assumptions have to be made about the texture. Assumptions such as directional isotropy and uniform density have been employed in previous research. Uniform density has been defined as density of texels or density of the sum of the lengths of the contours (zero-crossings) in the image.

It is very clear that even though some of the assumptions used in the literature for the recovery of shape from texture are general enough, they are not powerful enough to capture a very large subset of natural images. As a result, the developed algorithms fail when they are applied to many real surfaces. Furthermore, there is no way to check in advance whether or not a particular assumption is valid for the surface that is imaged. This problem alone is enough to demonstrate the restricted applicability of the existing shape from texture algorithms (or of the ones yet to come). We will show that if texture is combined with motion then the shape from texture problem, or the problem of shape detection from surface intensity and markings, becomes easy, in the sense that no restrictive assumptions are necessary and the solution is obtained from linear equations.

The next section introduces the mathematical prerequisites. For simplicity, and without loss of generality, we will assume that the surface in view is planar (as in previous shape from texture research); see Figure 5.1. If the surface in view is nonplanar, then the problem can be addressed either by applying our theory locally in the image, i.e. assuming that the surface in view is locally planar, or if a parametric model for the surface is assumed, then the same basic principles reported here may be used to recover the parameters of the surface (with the difference that the resulting equations might not be linear). We want a theory for shape from texture that would work for all kinds of images, such as the ones in Figures 5.2 5.4.

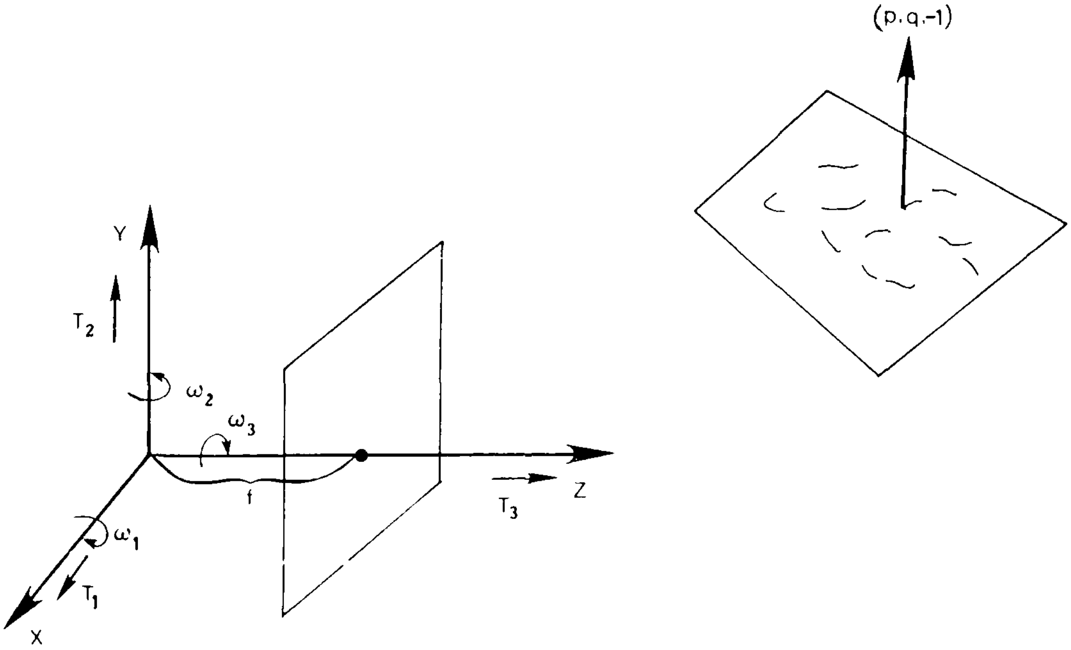


Figure 5.1: Motion of a textured planar surface.

Figure 5.2: Leaves.

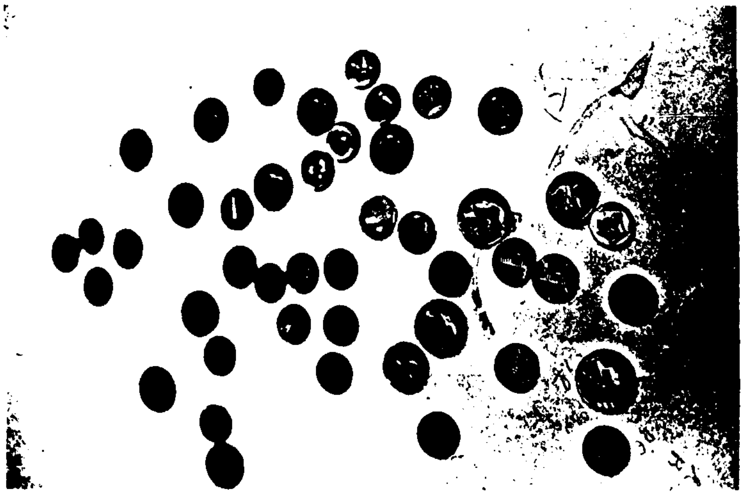


Figure 5.3: Coins.

5.1. Prerequisites

The treatment and symbolism here follow those in [Ito and Aloimonos, 1987]. Suppose that the camera is looking at a planar surface. Assume further that the camera is moving. For our analysis we assume that the surface is moving. This is equivalent to the motion of the camera, and it is done here for simplification of the formulas. Call the planar surface in the world W and the image plane R. Suppose that point $\mathbf{X} = (X,Y,Z) \in W$ is projected onto point $\mathbf{x} = (x,y) \in R$. Let the motion of the surface consist of a translation $\mathbf{T} = (t_1,t_2,t_3)$ and a rotation $\Omega = (\omega_1,\omega_2,\omega_3)$, or $\mathbf{V}(\mathbf{X}) = \mathbf{T} + \Omega \times \mathbf{X}$, where $\mathbf{V}(\mathbf{X})$ is the velocity of a point $\mathbf{X} \in W$. Then this velocity can be written as $\mathbf{V}(\mathbf{X}) = \sum_{k=1}^{6} r_k \mathbf{V}_k(\mathbf{X})$, where

$$r_1 = t_1, \quad V_1(\mathbf{X}) = (1 \ 0 \ 0)^T$$

$$r_2 = t_2, \quad V_2(\mathbf{X}) = (0 \ 1 \ 0)^T$$

$$r_3 = t_3, \quad V_3(\mathbf{X}) = (0 \ 0 \ 1)^T$$

$$r_4 = \omega_1, \quad V_4(\mathbf{X}) = (0 \ -Z \ Y)^T$$

$$r_5 = \omega_2, \quad V_5(\mathbf{X}) = (Z \ 0 \ -X)^T$$

$$r_6 = \omega_3, \quad V_6(\mathbf{X}) = (-Y \ X \ 0)^T$$

Then, it can be easily proved that the optic flow (image velocity) at a point $\mathbf{x} = (x,y)$ is $\dot{\mathbf{x}} = \sum_{k=1}^{6} r_k \, \mathbf{u}_k(\mathbf{x})$, where $\mathbf{u}_k$ is a function depending on the shape.

We prove the above equation avoiding the details of the perspective projection. Let the projection from the world to the image plane be $\mathbf{P}$, with $\mathbf{P}(\mathbf{X}) = \mathbf{x} = (x,y) = (\mathbf{P}_1(\mathbf{X}), \mathbf{P}_2(\mathbf{X}))$. If the shape of the surface W is given (a function h), a mapping: $\mathbf{P}_h^{-1}: R \rightarrow$ (object surface) is defined such that $\mathbf{P}(\mathbf{P}_h^{-1}(\mathbf{x})) = \mathbf{x}$.

The optic flow $\dot{\mathbf{x}}$ at a point $\mathbf{x} = (x,y)$ is then given by

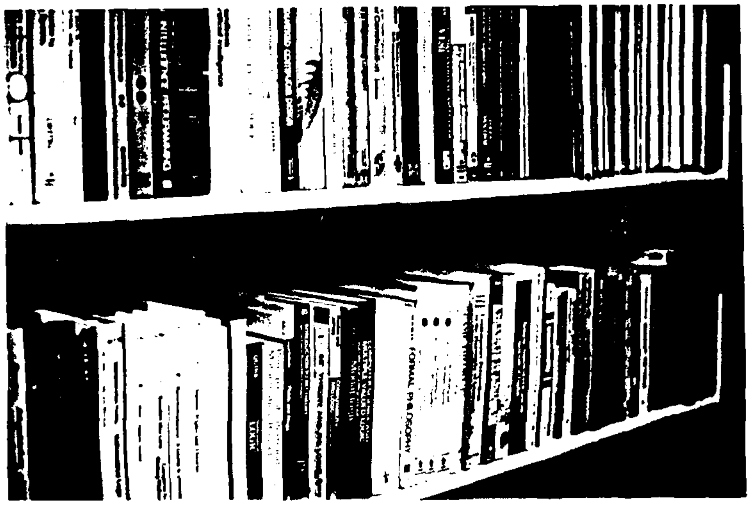


Figure 5.4: Books.

$$\dot{\mathbf{x}} = (\dot{x}, \dot{y}) = \frac{d}{dt}\mathbf{P}(\mathbf{X}) = \frac{\partial \mathbf{P}}{\partial \mathbf{X}}\mathbf{V}(\mathbf{X})$$

$$= \begin{bmatrix} \frac{\partial \mathbf{P}_1}{\partial X} & \frac{\partial \mathbf{P}_1}{\partial Y} & \frac{\partial \mathbf{P}_1}{\partial Z} \\ \frac{\partial \mathbf{P}_2}{\partial X} & \frac{\partial \mathbf{P}_2}{\partial Y} & \frac{\partial \mathbf{P}_2}{\partial Z} \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \\ V_3 \end{bmatrix}$$

Furthermore $\frac{\partial \mathbf{P}}{\partial \mathbf{X}}$ and $\dot{\mathbf{x}}$ are functions of $\mathbf{X}$, but using the inverse projection (from the image to the surface), we can consider them as functions of $\mathbf{x}$ (retinal coordinates). Thus, $\frac{\partial \mathbf{P}}{\partial \mathbf{X}} = \frac{\partial \mathbf{P}}{\partial \mathbf{X}}(\mathbf{P}_{h^{-1}}(\mathbf{x}))$ and $\mathbf{V} = \mathbf{V}(\mathbf{P}_{h^{-1}}(\mathbf{x}))$.

So, since $\mathbf{V} = \sum_{k=1}^{6} r_k \mathbf{V}_k$, we have

$$\dot{\mathbf{x}} = \sum_{k=1}^{6} r_k \mathbf{u}_k(\mathbf{x}) \text{ with} \tag{5.1}$$

$$\mathbf{u}_k(\mathbf{x}) = \frac{\partial \mathbf{P}}{\partial \mathbf{X}}(\mathbf{P}_{h^{-1}}(\mathbf{x})\mathbf{V}_k(\mathbf{P}_{h^{-1}}(\mathbf{x})).$$

Equation (5.1) will be used very frequently in the sequel.

5.2. Linear Features

Here we introduce the concept of a *linear feature vector* that has proved to be a strong device for several problems in cybernetics [Amari, 1986]. Let the image intensity function be denoted by $s(x,y)$. A *linear feature* (LF) is a linear functional f over the image, i.e.

$$f = \int\int s(x,y)m(x,y)dx\ dy,$$

where m is called a measuring function, s is the image brightness and the integration is taken over the area of interest. A *linear feature vector* $\mathbf{f}$ (LFV) is a vector of linear features, i.e.

$$\mathbf{f} = [f_1 f_2 \cdots f_n]^T, \text{ with}$$

$$f_i \quad \int\int s\ m_i\ dx\ dy$$

where m_i is a measuring function, for $i = 1, \ldots, n$. $\{m_i\}$ could be any set; one good example is $\{m_i\} = \{mpq\} = \{e^{i(px+qy)}\}$, in which case a linear feature corresponds to a Fourier component of the image.

It is clear that a linear feature is a global image feature, i.e. a global characteristic of the image. Using different measuring functions, we get different global measurements. The hope is that we might be able to connect properties of the 3-D surface in view through the concept of the linear feature and different positions of the camera. In that case we avoid any potential correspondences. The following section introduces the constraint that will be used later.

5.3. The Constraint

Since there is motion, the induced optical flow satisfies the following equation [Horn, 1986]:

$$s_x\ u + s_y\ v + s_t = 0,$$

where (u,v) is the optic flow at a point (x,y) and s_x, s_y s_t are the spatiotemporal derivatives of the image intensity function at the point (x,y). This equation can be written as

$$\frac{\partial s}{\partial t} = -\dot{\mathbf{x}} \cdot \nabla s.$$

The time derivative of an LFV will be

$$\dot{\mathbf{f}} = [\dot{f}_1 \dot{f}_2 \cdots \dot{f}_n], \text{ where}$$

$$\dot{f}_i = \int\int \frac{\partial s}{\partial t} m_i\ dx\ dy = -\int\int m_i(\dot{\mathbf{x}} \cdot \nabla s)dx\ dy$$

The optic flow field (from equation 5.1) can be written in the form

$$\dot{\mathbf{x}} = \sum_{k=1}^{6} r_k \mathbf{u}_k = \sum_{k=1}^{6} r_k \begin{bmatrix} u_k(\mathbf{x}) \\ v_k(\mathbf{x}) \end{bmatrix}$$

Up to this point we have connected the optic flow field with the parameters of the motion of our camera and the shape of the surface in view (functions $\mathbf{u}_k$). In the sequel, we will examine the temporal derivative of a global feature, i.e. how it changes with time.

We have

$$\dot{f}_i = -\int\int m_i(\dot{\mathbf{x}} \cdot \nabla s) dx\, dy \quad \text{or}$$

$$\dot{f}_i = -\sum_{k=1}^{6} r_k \int\int m_i(u_k s_x + v_k s_y) dx\, dy \quad \text{or}$$

$$\dot{f}_i = \sum_{k=1}^{6} r_k h_{ik}, \quad \text{with}$$

$$h_{ik} = -\int\int m_i(u_k s_x + v_k s_y) dx\, dy$$

So, we have found that

$$\dot{\mathbf{f}} = H \cdot r, \tag{5.2}$$

where $H = (h_{ik})$ and $r = (r_1 r_2 \cdots r_6)^T$, the motion parameters.

We have been rather abstract up to now, since we wished to emphasize that the theory does not depend on the projection used as well as the simplify formulae. With perspective projection (Fig. 5.1) the parameters in equation 5.1 are $r_1 = \frac{T_1}{c}$, $r_2 = \frac{T_2}{c}$, $r_3 = \frac{T_3}{c}$, $r_4 = \omega_1$, $r_5 = \omega_2$, $r_6 = \omega_3$, $u_1(\mathbf{x}) = (1 - px - qy, 0)$, $u_2(\mathbf{x}) = (0, 1 - px - qy)$, $u_3(\mathbf{x}) = (-x(1 - px - qy), -y(1 - px - qy))$, $u_4(\mathbf{x}) = (xy, y^2 + 1)$, $u_5(\mathbf{x}) = (-(x^2 + 1), -xy)$ and $u_6(\mathbf{x}) = (y, -x)$, where the motion is translation $T = (T_1, T_2, T_3)$ and rotation $\Omega = (\omega_1, \omega_2, \omega_3)$ and the plane in view has equation $Z = pX + qY + c$, with respect to the camera coordinate system (Fig. 5.1).

Matrix H contains the parameters of the plane in a linear form. So, equation (5.2) relates linear features with shape and motion parameters. Furthermore, it is linear in the shape of the planar surface in view. So, a simple linear least-squares method or a Hough transform technique is sufficient for the recovery of the gradient of the plane in view. Depth can be computed too, if desired. Here we must emphasize the fact that no local correspondence has been used. The only computed quantity was the time derivative of a linear feature vector, that involves the whole image.

Finally, we want to stress here the fact that in this algorithm, the spatial derivatives of the intensity function don't need to be computed. This is due to the linear feature vector approach. (Integration by parts avoids differentiation of the intensity function. Instead, the derivative of the measuring function has to be computed. So, we avoid differentiating the image intensity, which is discrete, because numerical differentiation is an ill-posed problem.) More importantly, the same approach can be followed if the image is a dot pattern (or a line pattern—zero crossings), i.e. it is discontinuous. The reason for this is again the fact that the spatial derivatives of the intensity function don't have to be estimated. Only the temporal derivative of the image needs be estimated. This approach has been initiated in [Ito and Aloimonos, 1987]. To further clarify, suppose that the image consists of a set of points (dot pattern), (x_i, y_i), $i = 1, \ldots, n$. Then, the moving image can be represented as $s(x,y,t) = \sum_{i=1}^{n} \delta(x - x_i(t), y - y_i(t))$. In that case a linear feature f_k is trivially computable. So, this theory works for both differential and nondifferential image functions. To summarize the algorithm for the detection of shape from texture from two images $s(x,y,t)$ and $s(x,y, t + dt)$ taken by a camera with known motion, the following are the successive steps of our formulation.

1. The input is a sequence of images $s(x,y,t)$ and $s(x,y, t + dt)$, taken with known motion.
2. Choose a set of measuring functions $\mu_i(x,y)$, $i = 1, \ldots, n$. These are smooth differentiable functions such as $x_i y_i$, $0 \leq i,j < k$, where k is a constant, or Fourier features such as $\cos(px + qy)$ where the coefficients p,q are determined in a random manner.
3. Create a linear feature vector

   $\mathbf{f} = [f_1 f_2 \cdots f_n]$ where

   $$f_i = \int\int s(x,y)\mu_i(x,y) dx\, dy,$$

   with the integration taken over the area of interest.
4. Compute the change of the linear feature vector $\mathbf{f}$, i.e. the time derivative $\dot{\mathbf{f}}$ from the images $s(x,y,t)$ and $s(x,y, t + dt)$.
5. Compute expressions for the entries of matrix H. For example:

   $$h_{13} = -\int\int m_i(u_3 s_x + v_3 s_y) dx\, dy \quad \text{or}$$

   $$h_{13} = -\int\int m_i(x(px + qy - 1)s_x + y(px + qy - 1)s_y) dx\, dy$$

6. From equation $\dot{\mathbf{f}} = H\, r$ solve for p,q (and/or) c with a least squares method.

5.4. Experiments

We have carried out experiments with synthetic and natural images. The pictures were processed by a VICOM processor and the motion was controlled using a Merlin American Robot arm. We will only present experiments with real images. Figure 5.5 shows the image of a drawing that was made with a marker on a white board. Figure 5.6 shows the image of the same scene after the motion of the camera. The orientation of the drawing with respect to the camera coordinate system was (ground truth) $p = 0.4$ and $q = 0.0$. It must be emphasized that estimating the ground truth must allow for an error of about 3-4 degrees in slant and tilt. The recovered parameters in this case were $p = .59$ and $q = -0.08$. Figures 5.7 and 5.8 show the images of a planar dragon (poster), before and after the motion, respectively. Again, the ground truth was $p = q = 0$ and the recovered shape was $p = 0.001$ and $q = 0.0003$.

Finally, the next experiment was designed in order to avoid the ambiguities that arise in the ground truth measurements. In this experiment we measure the change in shape, not the actual shape. Figures 5.9 and 5.10 show the images of a piece of cloth, before and after a small motion. Figures 5.11 and 5.12 show again the images of the cloth, after the camera has considerably rotated in order to change the orientation of

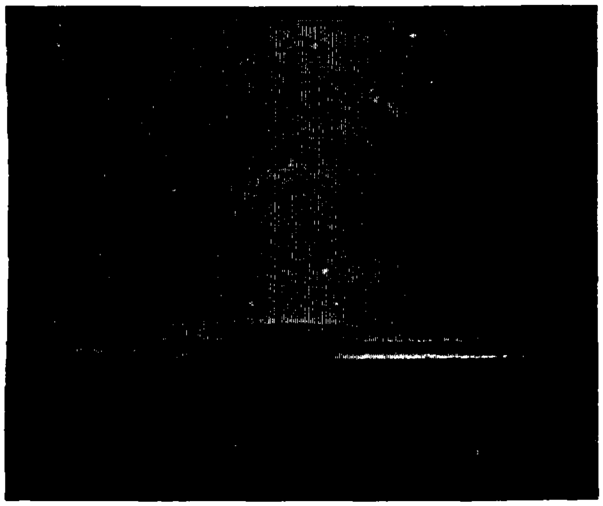


Figure 5.5: Drawing, before motion.

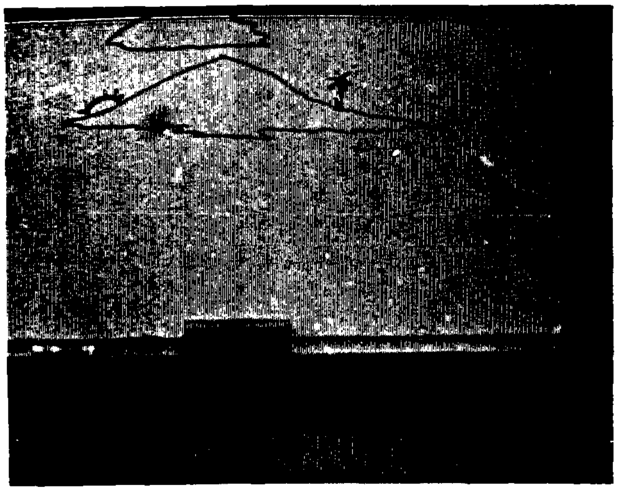


Figure 5.6: Drawing after motion.

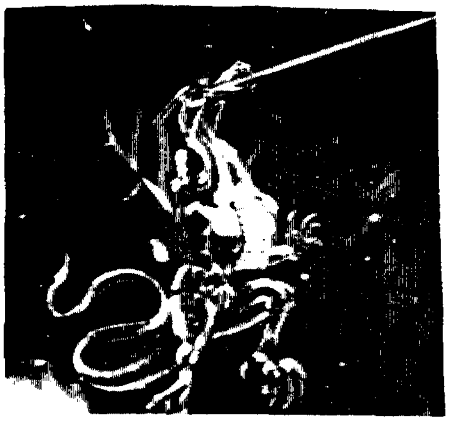


Figure 5.7: Dragon, before motion.

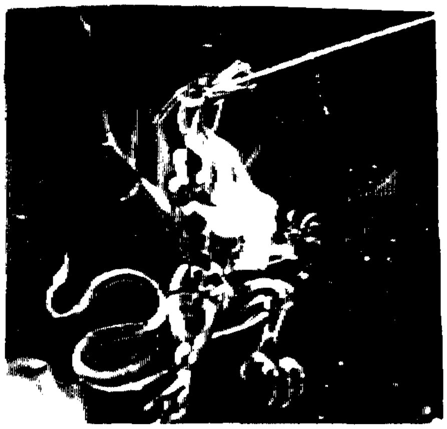


Figure 5.8: Dragon, after motion.

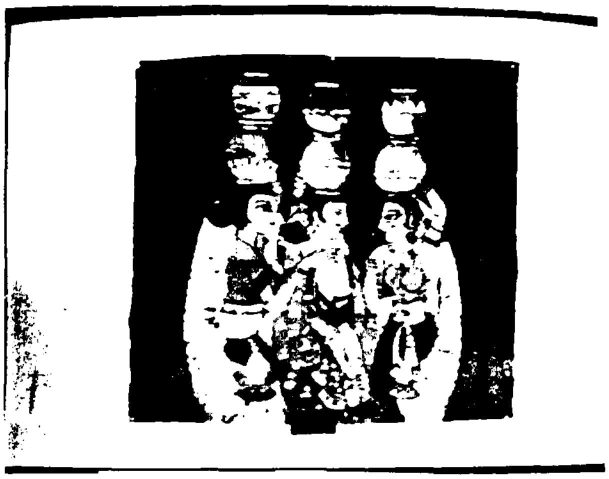


Figure 5.9: Cloth, before motion.

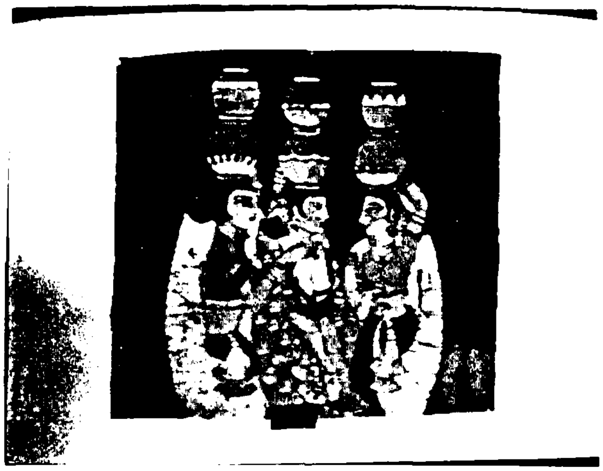


Figure 5.10: Cloth, after motion.

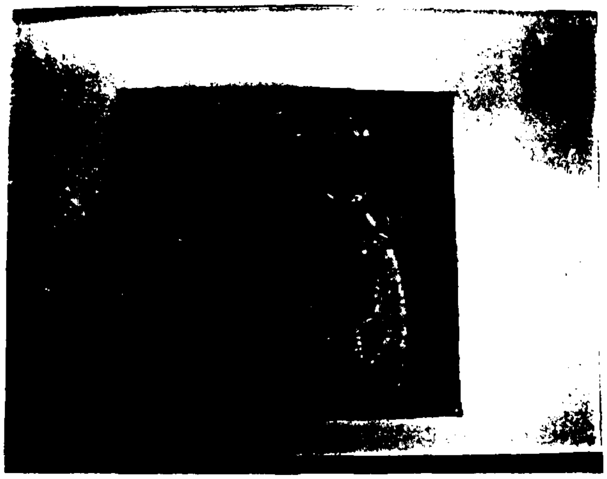


Figure 5.11: Cloth (rotated camera), before motion.

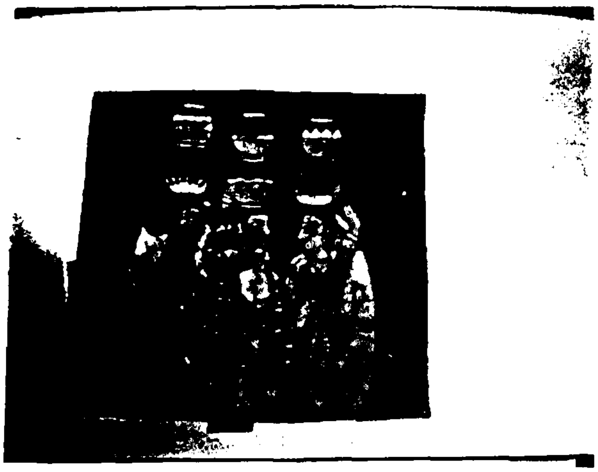


Figure 5.12: Cloth (rotated camera), after motion.

the cloth. We measure the two surface normals and then we compute the angle between the two normals, whose ground truth is known from the rotation of the camera. In this experiment the ground truth for the angle between the surface normals was $\theta = 15°$ and the estimated one was $\theta = 15.46°$.

5.5. Robustness Analysis

Equations (5.2) will finally result in a linear 3×3 system in the unknowns p,q and c of the parameters of the plane in view. It might happen, however, that this system, let's denote it by $A\vec{x} = \vec{c}$, where $\vec{x} = (p\ q\ c)^T$ and $A = (a_{ij})$ is a 3×3 matrix and c a 1×3 vector whose components are expressions of spatiotemporal derivatives of the image intensity functions. The fact that we just have to solve a simple linear system does not mean that its solution will be necessarily stable. Since there is discretization error as well as a slight error in the estimation of the known motion, there is some uncertainty in the entries of the matrix A and the vector c, the true values not being known exactly. If the true system corresponding to the above is $A^* x^* = c^*$, let's suppose that the coefficients and the right hand constants of the true system of equations are known no more precisely than is given by

$$a_{ij}^* \in [a_{ij} - \epsilon_{ij}, a_{ij} + \epsilon_{ij}]$$

and

$$c_i^* \in [c_i - \epsilon_i, c_i + \epsilon_i]$$

where the quantities ϵ_{ij}, ϵ_i are nonnegative. We call these quantities uncertainties. If there exist values of the coefficients a_{ij} in the intervals of uncertainty for which the determinant of the system becomes zero, then the system is very badly conditioned and it shouldn't be solved, because whatever the solution is, it will be very unreliable. In the sequel we will develop a condition for this kind of ill-conditionedness to occur. Let's first write the true system of equations $A^* \vec{x}^* = \vec{c}^*$ as

$$(A + \delta A)(\vec{x} + \delta\vec{x}) = \vec{c} + \delta\vec{c}$$

where A, $\vec{x}$ and $\vec{c}$ refer to the approximate system of equations. Let's also write $\Delta A = (\epsilon_{ij})$ and $|\delta A| = (|\delta a_{ij}|)$ where $\delta A = (\delta a_{ij})$. Then in order to have stability it is necessary and sufficient that $A + \delta A$ be nonsingular for changes in the coefficients within the limits of the uncertainties, i.e. we must have

$$\det(A + \delta A) \neq 0 \tag{5.3}$$

whenever

$$|\delta A| \leq \Delta A$$

i.e. whenever $|\delta a_{ij}| \leq \epsilon_{ij}$. But the determinant is a function of the coefficients. So, writing $a = \det(A)$ and $a + \delta a = \det(A + \delta A)$, the change δa in the determinant a is given to the first order of small quantities by

$$\delta a = \sum_{i=1}^{n} \sum_{j=1}^{n} \frac{\partial a}{\partial a_{ij}} \delta a_{ij} \tag{5.4}$$

Then, taking the modulus we have

$$|\delta a| \leq \sum_{i=1}^{n} \sum_{j=1}^{n} \left| \frac{\partial a}{\partial a_{ij}} \right| |\delta a_{ij}|$$

and the equality is reached in the above equation when $\frac{\partial a}{\partial a_{ij}}$ and δa_{ij} are always of the same sign.

Let's further denote by Δa *the least upper bound of the* absolute change in value of the determinant within the limits of the uncertainties in the coefficients. Thus, within the limits *a change in value of the determinant* as large as Δa in absolute value is possible, but a larger value is not possible. So,

$$\Delta a = \sum_{i=1}^{n} \sum_{j=1}^{n} \left| \frac{\partial a}{\partial a_{ij}} \right| \epsilon_{ij}$$

The expression on the right hand side of the above equation is an upper bound of $|\delta a|$ for $|\delta a_{ij}| \leq \epsilon_{ij}$, and this expression is the least upper bound. Now, instability (critical ill-*conditioning) means that the determinant of the coefficient* matrix can become zero within the limits of the uncertainties in the coefficients. But the true value of the determinant, from the preceding analysis, is contained in the interval $[a - \Delta a, a + \Delta a]$, and so the necessary and sufficient condition for no critical ill-conditioning is that this interval does not contain zero, i.e. $\Delta a < |a|$, whatever the sign of a. So, the necessary and sufficient condition for no critical ill-conditioning is

$$\Delta a < |a|$$

or

$$\sum_{i=1}^{n}\sum_{j=1}^{n}\left|\frac{\partial a}{\partial a_{ij}}\right|\epsilon_{ij} < |a|$$

On the other hand, we have

$$\frac{\partial a}{\partial a_{ij}} = (-1)^{i+j} M_{ij}$$

for $i,j = 1, \ldots, 3$, where M_{ij} are the minors of the determinant of the matrix A. But it is trivial to see that

$$\frac{(-1)^{i+j} M_{ij}}{a}$$

are the elements of the transpose of the inverse of A, i.e.

$$b_{ji} = \frac{(-1)^{i+j} M_{ij}}{a} \quad i,j = 1,2, \ldots, 3$$

where $B = (b_{ji}) = A^{-1}$.

Hence, from this analysis, the necessary and sufficient condition for the system not to be critically ill-conditioned (and therefore for its solution to be stable) is

$$\sum_{i=1}^{n}\sum_{j=1}^{n}|b_{ji}|\epsilon_{ij} < 1 \tag{5.5}$$

It has to be noted, however, that this condition is approximate, since we assumed that (5.4) is true (first order); but this approximation is not very far from reality, since the determinant is a *smooth function of the elements of the matrix*. We now proceed to apply the findings of this section to our problem. A theoretical error analysis seems very hard, since we would need specific distributions for the image intensity function. However, we can examine if (5.5) becomes true for several instances of our problem.

Introducing discretization error (i.e. retinal coordinates x,y have uncertainties of at most 1/2 pixel), allowing small errors (up to 20%) in the known motion parameters, and generating random intensity images, we found from simulation that condition 5.5 was always true, i.e. we never had critical ill-conditioning as also the previous experimental results indicated. However, a theoretical error analysis is still needed.

5.6. Summary and Discussion

We have presented a method for the recovery of shape. Our method does not rely on any assumptions about the texture and it does not require the image to be spatially differentiable. The approach is based on the fact that the observer is moving with a known motion. If the observer is moving with an unknown motion, then again the problem is solvable, and it has been addressed by Aloimonos [1986] and Neghadaripour [1985].

6. SHAPE AND 3-D MOTION FROM CONTOUR AND STEREO

In this section we study the detection of surface shape and three-dimensional motion from the perception of a planar contour. We prove that a binocular observer can compute the orientation and the 3-D motion of a moving contour without using point to point correspondences. In particular:

(1) We develop constraints between the coordinates of the points that constitute the contours in the left and right retina of a binocular observer that enable him to detect the structure and depth of the plane in view without using any point to point correspondences.

(2) We develop constraints between the lengths and the areas of the contours in the left and the right retina of a binocular observer that enable him to compute the structure and the depth of the plane in view without any point correspondences.

(3) We discover constraints between the retinal motion of the contour and its three-dimensional motion that make it possible to recover 3-D motion without any correspondences.

(4) And finally we generalize some of the above results for a monocular observer. In particular, a translating monocular observer can recover the shape of an imaged contour without using any point to point correspondences.

The basic assumption here is that the contours in the left and right images have been found and the correspondence between them has been established.

6.1. Introduction

The human perceiver is able to derive enormous amounts of information from the contours in a scene. As part of this capacity, we are able to use the shapes of image contours (as they are seen by both eyes) to infer the shapes and dispositions in space of the surfaces they lie on, as well as their motion. To the extent the inferences we draw are accurate, our strategies for drawing them must have some basis in the character of the visual world, just as the efficacy of stereopsis as a source for depth information has a basis in the geometry of projection and triangulation. The aim of the research described here is (1) *to discover constraints on the visual world that allow surface shape and motion to be reliably inferred from contours in images,* (2) *to derive methods of inference from these constraints.* The interpretation of contours by a binocular observer falls into four subproblems (following [Witkin, 1981]). In particular these four subproblems are the following:

a) *Locating contours in the images.*

If contours are to be used to infer anything, they must be found. The human perceiver has little difficulty deciding what is and is not a contour, yet the automatic detection of edges has proved very difficult. Perhaps this fact should not be surprising; the contours that we see in natural images usually correspond to definite physical events, such as shadows, depth discontinuities, color differences and the like. Our ability to detect these events may say more about their significance for image interpretation than about their ease of detection. Why should we expect events that have simple descriptions in terms of the structure of the scene to have simple descriptions in terms of the image intensity as well? If the physical significance of contours is taken as their primary feature, then at least we know what is being detected, even if we don't know how. But recent research shows that we are in pretty good state as far as detection of contours goes. Actually, we can say that we can fairly well detect the contours in an image, even if there are some inaccuracies.

b) *Labeling contours (i.e. distinguishing contours which are due to different physical events)*

If contours correspond to different physical events, then an essential component of their interpretation must be to decide which contours denote which event, since each kind of contour imparts a different meaning. Recent work has shown that strong structural constraints can be applied to distinguish one kind of contour from another.

c) *Corresponding contours (i.e. finding which contours in the left and right images are the image of the same 3-D contour)*

Before we apply some interpretation method to the images of the contours (left and right), we should know which contours in both images correspond to each other, i.e. they are the images of the same three-dimensional contour.

d) *Interpreting contours*

Even after contours have been found, labeled and the corresponding ones in the left and right images have been identified, not much is known about the physical structure of the scene, if we don't wish to resolve in a point-to-point correspondence between the left and right images. It is clear that contours play an important role in the human perceiver's ability to decide how things are shaped and where they are, apart from the application of specific "higher level" knowledge to objects of known shape. This research addresses this fourth problem, i.e., given the left and right image of a moving planar 3-D contour, to recover its orientation, depth and 3-D motion, without using any point-to-point correspondence, neither between the left and right images nor between the dynamic frames. The reason that we want to solve the problem without using point correspondences is that correspondence is a very hard problem and it does not seem tractable with the available tools. So, we would like to address the problem in such a way that we avoid the correspondence problem.

6.2. Motivation

This research is motivated by the inherent difficulties of the conventional stereo problems as well as the difficulty of the dynamic correspondence problem (to recover optic flow or discrete displacements, that will be used for the recovery of 3-D motion).

Passive ranging by triangulation methods, which is employed successfully by humans under certain conditions, has received much attention in the computer vision literature in recent years. It is obvious that the ability to recover absolute range to objects in a scene would be important in a variety of robotic applications. To date, only two basic methods of passive ranging have been reported, "static stereo," i.e. the use of two cameras separated by a known baseline, and "motion stereo," i.e. the use of a single camera moving in a known way through a stationary scene. Recently, a new concept has been introduced for passive ranging to moving objects, termed "dynamic stereo," which is based on the comparison of multiple image flows. In the sequel, we will only deal with the criticism of the first method (static stereo). Most of the literature on passive ranging has been concerned with the difficult "correspondence" problem associated with the assignment of stereo disparities (for the static stereo method). Beside the traditional method of intensity correlation between images, much attention has been paid to the theory of Marr and Poggio [1977, 1979], with implementation by Grimson. The use of more than two camera locations, to aid in solving the correspondence between images, has been approached in different ways by some researchers. Nevertheless, the solution of this correspondence problem remains a computationally expensive and slow process, with partial success in a variety of input images. Moreover, a maximum ranging distance is implied by the finite resolution of the cameras and the statically configured baseline between cameras. Most of the work needed to solve the correspondence problem deals with the matching of microfeatures, such as points of interest (corners, high curvature points) and edges. A natural questions that arises, then, is: Is it possible to recover structure and depth, given that we have matched a macrofeature (i.e., a planar contour) instead of a microfeature? We prove that it is. Of course in this study we don't deal with "how to match the planar contours in the two stereo frames," i.e., to find in both images the contours which are due to the projection of the same three-dimensional planar contour (size, color, texture, fractal dimension could be used for the solution of this problem).

We also show that it is possible to solve the 3-D motion determination problem without using point-to-point correspondence for the case where the image object is a planar contour. Here, we show that it is possible for the case of a planar contour, i.e., a binocular observer can understand the 3-D motion of a contour, from two temporally close positions of the contour, without using any point-to-point correspondence. Of course there are still difficulties with this new approach and the inherent problems of the dynamic imagery appear in another form, different from the one in the traditional methods (camera ⇛ retinal motion ⇛ 3-D motion); but it turns out that these problems, in the presence of small noise percentages, hardly affect the results.

The organization of the section is as follows. Section 6.3 describes previous work. Section 6.4 introduces the concept of "aggregate stereo," a method that computes the structure and depth of a 3-D planar contour from its images on the left and right flat retina. Section 6.5 introduces new constraints for the stereo problem, which are not based on triangulation, but on the change of area and perimeter in the left and right images of the contour. Section 6.6 introduces the concept of determining the direction of the translation of a translating planar contour, without using any point-to-point correspondence, and introduces the reader to Section 6.7 which deals with the solution of the general problem (the case where the 3-D planar contour is translating and rotating).

In what follows, because of the discrete nature of images, we will consider a contour either as a collection of points (which it actually is) or as a continuous curve, when needed to establish the mathematical rigorousness of a proof.

6.3. Previous Work

The idea of using more than one camera to recover the shape of a contour seems to be new.

The recovery of three-dimensional shape and surface orientation from a two-dimensional contour is a fundamental process in any visual system. Recently, a number of methods have been proposed for computing this shape from contour. For the most part, previous techniques have concentrated on trying to identify a few simple, general constraints and assumptions that are consistent with the nature of all possible objects and imaging geometries in order to recover a single "best" interpretation, from among the many possible for a given image. For example, Kanade [1981] defines shape constraints in terms of image space regularities such as parallel lines and skew symmetries under orthographic projection. Witkin [1981] looks for the most uniform distribution of tangents to a contour over a set of possible inverse projections in object space under orthography. Similarly, Brady and Yuille [1984] search for the most compact shape (using the measure of area over perimeter squared) in the object space of inverse projected planar contours.

Rather than attempting to maximize some general shape-based evaluation function over the space of possible inverse projective transforms of a given image contour, and keeping in our framework of attempting unique solutions without employing any restrictive assumptions and heuristics, we propose to find a unique solution by using more than one camera, since it can be easily proved that only one image (under orthography or perspective) of a planar contour admits infinitely many interpretations of the structure of the world plane on which the contour lies, if no other information is known. Finally, the need for a unique solution, which is guaranteed in our approach, comes also from the fact that there exist many real world counterexamples to the evaluation functions that have been developed to date. For example, Kanade's and Witkin's measures incorrectly estimate surface orientation for regular shapes such as ellipses (which are often interpreted as slanted circles). Brady's compactness measure does not correctly interpret non-compact figures such as rectangles since he will compute them to be rotated squares (e.g. if we view a rectangular table top, we do not see it as a rotated square surface, but as a rotated rectangle).

Finally, the need for the solution of the 3-D motion parameter determination problem without using point-to-point correspondence for the case of the planar contour has recently been appreciated by Kanatani [1985], and has led to methods of great mathematical elegance. The methods that we will propose in the following sections are quite intuitive and can be considered immune to small noise percentages.

6.4. Aggregate Stereo

In this section we present a theory for the recovery of the three-dimensional parameters of a planar contour, from its left and right images, without using any point-to-point correspondence. Instead, we consider all the point correspondences at once; thus, there is no need for the solution of the correspondence problem of points. Correspondence of the contours as a whole is required. We consider a contour to be discrete, i.e. a collection of points.

Let a coordinate system $OXYZ$ be fixed with respect to the left camera, with the Z axis pointing along the optical axis. We consider that the image contour is a collection of points. So, $C_l = \{(x_{li},\ y_{li}) | i = 1, \dots, n\}$ and $C_r = \{(x_{ri},\ y_{ri}) | i = 1, \dots, n\}$. Consider a point $(X_i,\ Y_i,\ Z_i)$ on the world plane and its projections $(x_{li},\ y_{li})$, $(x_{ri},\ y_{ri})$ on the left and right image frames respectively. Then

$$x_{li} - x_{ri} = \frac{fd}{Z_i} \tag{6.1}$$

$$y_{li} = y_{ri} \tag{6.2}$$

with f the focal length (see Fig. 6). We proceed with the following propositions.

Proposition 1: Under the established nomenclature, the quality

$$\sum_{i=1}^{n} \frac{y_{li}^{k}}{Z_i}$$

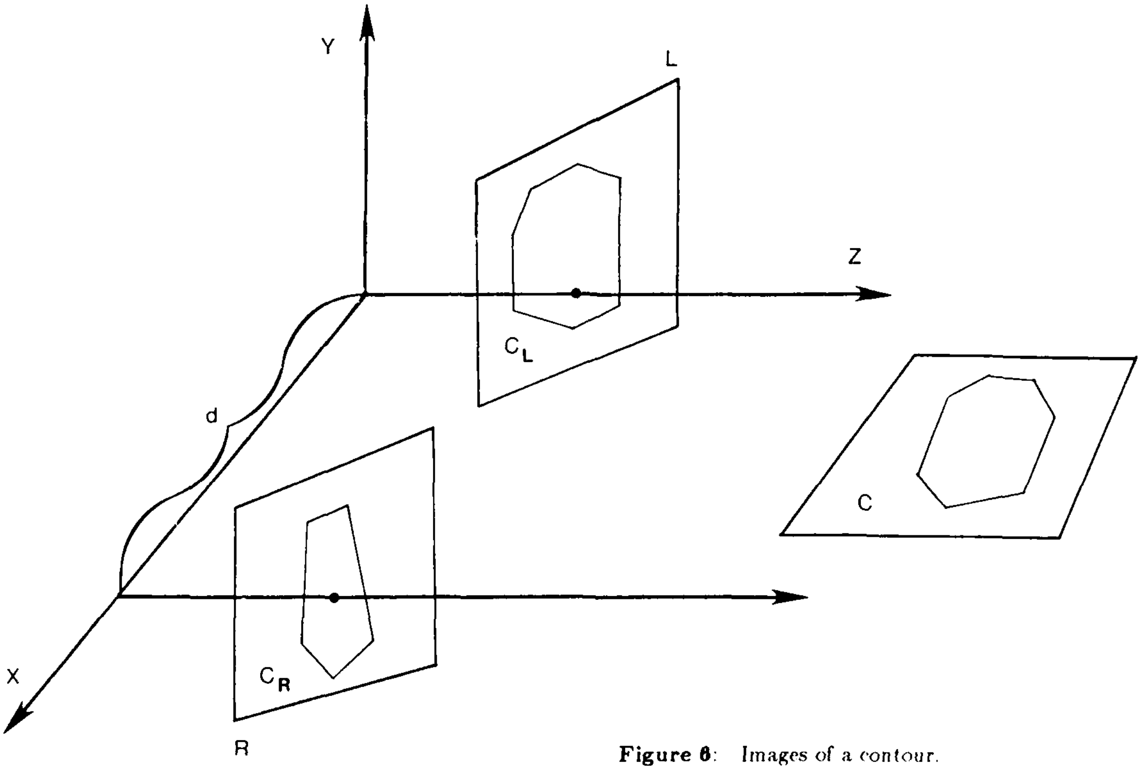


Figure 6: Images of a contour.

is directly computable (no correspondence is required).

Proof: We have

$$\sum_{i=1}^{n} \frac{y_{li}^{k}}{Z_i} = (\text{from equation } (6.1)) = \sum_{i=1}^{n} y_{li}^{k} \frac{(x_{li} - x_{ri})}{f^{*} d}$$

$$= \sum_{i=1}^{n} \frac{x_{li}\ y_{li}^{k}}{fd} - \sum_{i=1}^{n} \frac{x_{ri}\ y_{li}^{k}}{fd} = \ (\text{from equation } (6.2))$$

$$= \sum_{i=1}^{n} \frac{x_{li}\ y_{li}^{k}}{fd} - \sum_{i=1}^{n} \frac{x_{ri}\ y_{ri}^{k}}{fd}.$$

Thus,

$$\sum_{i=1}^{n} \frac{y_{li}^{k}}{Z_i} = \sum_{i=1}^{n} \frac{x_{li}\ y_{li}^{k}}{fd} - \sum_{i=1}^{n} \frac{x_{ri}\ y_{ri}^{k}}{fd} \tag{6.3}$$

From equation (6.3) the claim is obvious.

Proposition 2: Using the aforementioned nomenclature, the parameters p,q and c of the contour in view are directly computable without using any point-to-point correspondence between the two frames.

Proof: The equation of the world planar contour when expressed in terms of the coordinates of the left frame becomes

$$\frac{1}{Z} = (f - px_l - qy_l)\frac{1}{cf} \tag{6.4}$$

It follows that

$$\frac{1}{Z_i} = (f - px_{li} - qy_{li})\frac{1}{cf} \quad i = 1, 2, 3, \ldots, n \tag{6.5}$$

Now, we have

$$\sum_{i=1}^{n} \frac{y_{li}^{k}}{Z_i} = \sum_{i=1}^{n} (f - px_{li} - qy_{li}) \frac{y_{li}^{k}}{cf}$$

or

$$\sum_{i=1}^{n} \frac{y_{li}^{k}}{Z_i} = \frac{1}{c} \sum_{i=1}^{n} y_{li}^{k} - \frac{1}{cf} \left[\sum_{i=1}^{n} px_{li} y_{li}^{k} + \sum_{i=1}^{n} qy_{li} y_{li}^{k} \right] \tag{6.6}$$

The left-hand side of equation (6.6) has been shown to be computable without using any point-to-point correspondence (see Proposition 1).

If we write equation (6.6) for three different values of k, we obtain the following linear system in the unknowns p, q, c which in general has a unique solution:

$$\sum_{i=1}^{n} \frac{x_{li}\ y_{li}^{k1}}{fd} - \sum_{i=1}^{n} \frac{x_{ri} y_{ri}^{k1}}{fd} = \frac{1}{c} \sum_{i=1}^{n} y_{li}^{k1} - \frac{1}{cf} \left[\sum_{i=1}^{n} px_{li} y_{li}^{k1} + \sum_{i=1}^{n} qy_{li} y_{li}^{k1} \right] \tag{6.7}$$

$$\sum_{i=1}^{n} \frac{x_{li} y_{li}^{k2}}{fd} - \sum_{i=1}^{n} \frac{x_{ri} y_{ri}^{k2}}{fd} = \frac{1}{c} \sum_{i=1}^{n} y_{li}^{k2} - \frac{1}{c^{*} f} \left[\sum_{i=1}^{n} px_{li} y_{li}^{k2} + \sum_{i=1}^{n} qy_{li} y_{li}^{k2} \right] \tag{6.8}$$

$$\sum_{i=1}^{n} \frac{x_{li} y_{li}^{k3}}{fd} - \sum_{i=1}^{n} \frac{x_{ri} y_{ri}^{k3}}{fd} = \frac{1}{c} \sum_{i=1}^{n} y_{li}^{k3} - \frac{1}{c^{*} f} \left[\sum_{i=1}^{n} px_{li} y_{li}^{k3} + \sum_{i=1}^{n} qy_{li} y_{li}^{k3} \right] \tag{6.9}$$

where we have used equation (6.3) in the left hand sides.

The solution of the above system recovers the structure and the depth of the world planar contour without any correspondence and this is the conclusion of Proposition 2.

6.4.1. Practical considerations

It is clear from the previous analysis that we have considered that the number of points in the left and right contours (number of points in the contours on the left and right image plane) is the same. But in a practical situation, depending on the orientation of the world contour and due to foreshortening effects, the number of points in the left and right image contours will not be the same. To account for this, we divide the summations in equations (6.7), (6.8) and (6.9) by the total number of points. For example, one of these equations becomes

$$\frac{c}{fd}\left(\frac{1}{n_l} \sum x_{li} (y_{li})^{k} - \frac{1}{n_r} \sum x_{ri} (y_{ri})^{k} \right)$$
$$= f \frac{1}{n_l} \sum (y_{li})^{k} - p\frac{1}{n_l} \sum x_{li} (y_{li})^{k} - q\frac{1}{n_l} \sum y_{li}^{k+1} \tag{6.10}$$

where n_l, n_r are the total numbers of points constituting the left and right image contours respectively. To check the consistency of these equations (unbiasedness, sufficiency), we assume that the points are perturbed, i.e.

$$x_{li} = X_{li} + \epsilon(x_{li}) \text{ with } \epsilon(x_{li}) \sim N(0.\sigma^2)$$

and the same for the rest of the coordinates, with the assumption of independence among the stochastic variables x_{li}, y_{li}, x_{ri} and y_{ri}. We will prove in the sequel that equation (6.10) is consistent if and only if $k = 0$.

Indeed, consider

$$E\left(\frac{1}{n_l} \sum x_{li} (y_{li})^{k} \right) = \frac{1}{n_l} \sum E(x_{li}) E(y_{li}^{k})$$

$$= \frac{1}{n_l} \sum X_{li}\ E(y_{li}^{k}) = \frac{1}{n_l} \sum X_{li}\ Y_{li}^{k}$$

iff $k = 0$ or 1, since $E(y_{li}^{k}) = Y_{li}^{k}$ iff $k = 0$ or 1. But

$$E\left(\frac{1}{n_l} \sum (y_{li})^{k+1} \right) = \frac{1}{n_l} \sum (Y_{li})^{k+1}$$

iff $k = -1$ or 0.

So, for unbiasedness of error it is necessary that $k = 0$. To check sufficiency we need to check whether or not

$$\text{Var}\left(\frac{1}{n_l} \sum x_{li} \right) \to 0 \text{ as } n_l \to \infty.$$

But $\text{Var}\left(\frac{1}{n_l} \sum x_{li} \right) = \frac{1}{n_l^{2}} \sum \text{Var}(x_{li})$

$$= \frac{1}{n_l^{2}} n_l \sigma^2 = \frac{\sigma^2}{n_l} \to 0 \text{ as } n_l \to \infty.$$

Similarly for the other coefficients. Thus, in order to have a robust method, we showed that the exponents k should be zero, in which case all three equations (6.7), (6.8) and (6.9) degenerate to one equation. In that case, we need to divide the contours into three separate parts (see Figure 6.1) and apply equation 6.10 to each one of the corresponding parts. It can be easily shown that the resulting system of three equations in the three unknowns p, q, c always has a solution, unless the centers of mass of the three different areas are collinear (Fig. 6.1).

6.5. Shape from Contour and Stereo Based on Change of Length and Area

In this section we develop a computational theory for the detection of shape from contour by a binocular observer, without using correspondences, and utilizing invariant properties of area and length.

Consider a coordinate system $OXYZ$ to be fixed with respect to the left camera, with the $-Z$ axis again pointing along the optical axis. We consider that the image plane of the left camera is perpendicular to the Z axis at the point (0,0,1). The nodal point of the right camera is the point $(\Delta x,0,0)$ and the image plane of the right camera is identical to the one of the left camera. C is a contour on the world plane Π with equation $Z = pX + qY + c$, and C_l and C_r are the projections of the contour C on the left and right image respectively, using perspective projection (see Figure 6).

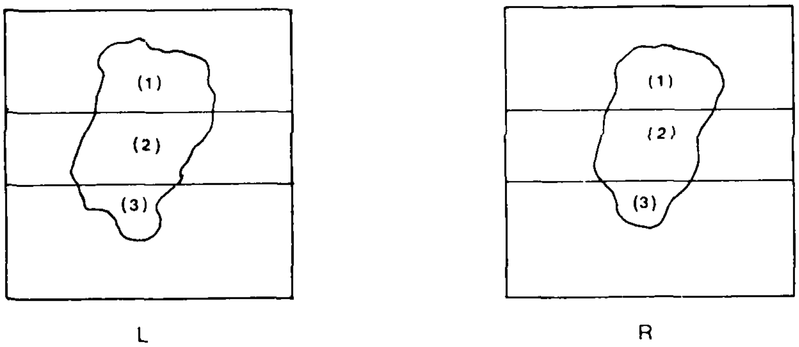


Figure 6.1: Dividing the contours into three parts.

There exist invariant quantities in this particular case, namely properties of the 3-D contour that can be found from either frame (left or right). As such properties, we choose area and perimeter. In other words, the change of the area of the contour from the left to the right frame as well as the change in the perimeter should be connected to intrinsic properties of the world contour, for example, its orientation. If S_L and S_R are the areas of the image contours in the left and right images respectively, then the following relation holds:

$$\frac{S_L}{S_R} = \frac{1 - A_L p - B_L q}{1 - A_R p - B_R q}, \qquad (6.11)$$

where (A_L, B_L), (A_R, B_R) the centers of mass of the left and right image contours respectively and the focal length is unity. For the proof of (6.11) see Appendix 2.

Clearly, (6.11) is a linear equation in the unknowns p,q, i.e. the gradient of the world plane. On the other hand, there is information from the perimeters of the image contours and we haven't yet used it. In particular, if we compute the perimeter of the world contour from the left image or from the right image, the result should be the same. From this fact we can derive one more constraints and thus solve for the orientation of the surface in view.

For that, we need to develop the first fundamental form of the world plane as a function of the retinal coordinates, in order to be able to compute the length of the world contour (up to a constant factor, of course). If we fix a coordinate system $OXYZ$ with the Z axis as the optical axis and focal length $\mathbf{F} = 1$ and we consider a plane Π: $Z = pX + qY + c$ in the world with a contour C on it, and we denote by (x,y) the coordinates on the image plane, then a point (X,Y,Z) in the world planar contour C is projected onto the point

$$x = \frac{X\mathbf{F}}{Z};\ y = \frac{Y\mathbf{F}}{Z}$$

The inverse imaging function, call it f, is the function that maps the image plane onto the world plane; so, if (x,y) is an image point, the 3-D world point in the plane $Z = pX + qY + c$ that has (x,y) as its image is given by

$$f(x,y) = \left(\frac{cx}{\mathbf{F} - px - qy}, \frac{cy}{\mathbf{F} - px - qy}, \frac{c\mathbf{F}}{\mathbf{F} - px - qy} \right)$$

The first fundamental form of f [Lipschutz, 1969] is the quadratic form

$$E\ dx^2 + 2F\ dx\ dy + E\ dy^2$$

with

$$E = f_x \cdot f_x$$
$$F = f_x \cdot f_y \text{ and}$$
$$G = f_y \cdot f_y$$

or

$$E = \frac{c^2}{(1 - px - qy)^4}[(1 \quad qy)^2 + p^2 y^2 + p^2]$$

$$F = \frac{c^2}{(1 - px - qy)^4}[(1 - qy)qx + (1 - px)py + pq]$$

$$G = \frac{c^2}{(1 - px - qy)^4}[q^2 x^2 + (1 - px)^2 + q^2]$$

If we consider two points (x,y) and $(x + dx,\ y + dy)$ on the image plane, then the 3-D distance dC of the corresponding points on the world plane is given by

$$dC = \sqrt{E\ dx^2 + 2F\ dx\ dy + G\ dy^2}$$

Consequently, if we have a contour C on the image plane, then the 3-D planar contour has perimeter length

$$\int_C \sqrt{E\ dx^2 + 2F\ dx\ dy + G\ dy^2} \text{ on the image plane} \qquad (6.12)$$

Expression (6.12) can be used to compute the length of the world contour from either image frame. Let L_l and L_r the expressions for the length of the 3-D contour as computed from the left and right image frames respectively. Thus,

$$L_l - L_r = 0 \qquad (6.13)$$

represents a nonlinear constraint on the parameters of the world plane. It has to be emphasized that equation (6.13) involves as unknowns only the components of the gradient (p,q) of the plane in view; the constant c is eliminated using an approximation of the perspective projection, called paraperspective [Aloimonos, 1986]. The system of equations (6.11) and (6.13) gives a finite number of solutions for the orientation of the contour (p,q).

6.5.1. Practical considerations

A closed form solution for the system of equations (6.11) and (6.13) appears very hard, if not impossible, since no specific contours are assumed. Changing the formulation for surface orientation from gradient space to the Gaussian sphere [Horn, 1986], we need to compute the azimuth θ and elevation ϕ of the planar surface in view. Equation (6.11) when transformed to the coordinates of the Gaussian sphere represents a great circle G. To solve the problem, we need to find which point of that great circle makes equation (6.13) correct. In other words, in a discretized Gaussian sphere we need to find the point $(\theta,\phi) \in G$ that minimizes the function $(L_l - L_r)^2$. Multiple solutions are possible even though our experiments showed unique solutions for the examples considered. Figure 6.1.1 shows the constraints from area and length drawn on the Gaussian sphere. It was recently shown [Aloimonos and Hervé, 1987] that there can be at most two solutions; and criteria for checking multiplicity of the solutions have been developed.

6.6. Determining the Direction of Translation

Here we only treat the case of pure translation. The general case is treated in the next section. The treatment in this section presumes perspective projection.

Consider a coordinate system $OXYZ$ fixed with respect to the camera, O the nodal point of the eye and the image plane perpendicular to the Z axis (focal length 1) that is pointing along the optical axis. Let us represent points on the image plane with small letters $((x,y))$ and points in the world with capital letters $((X,Y,Z))$. (See Figure 6.2.)

Let a point $P = (X_1, Y_1, Z1)$ in the world have perspective image (x_1, y_1) where $x_1 = X_1/Z_1$ and $y_1 = Y_1/Z_1$. Let the point P move to the position $P' = (X_2, Y_2, Z_2)$ with

$$X_2 = X_1 + \Delta X$$
$$Y_2 = Y_1 + \Delta Y$$
$$Z_2 = Z_1 + \Delta Z,$$

Then we desire to find the direction of the translation $(\Delta X/\Delta Z,\ \Delta Y/\Delta Z)$. If the image of P' is (x_2, y_2) then the observed motion of the world point in the image plane is given by the displacement vector $(x_2 - x_1,\ y_2 - y_1)$ (which in the case of very small motion is also known as optic flow).

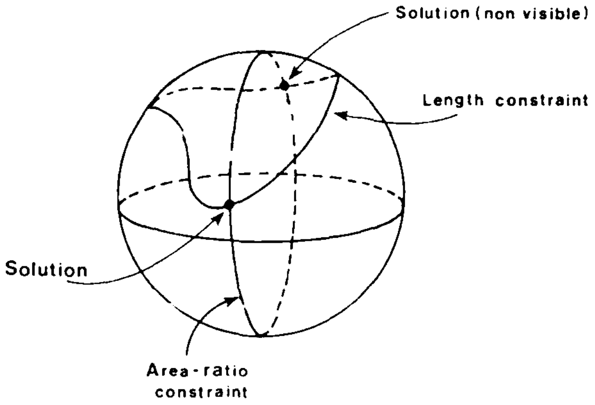


Figure 6.1.1: Length and area constraints.

We can easily prove that

$$x_2 - x_1 = \frac{\Delta X - x_1 \Delta Z}{z_1 + \Delta Z}$$

$$y_2 - y_1 = \frac{\Delta Y - y_1 \Delta Z}{z_1 + \Delta Z}$$

Under the assumption that the depth is large (and the motion in depth small), the equations above become

$$x_2 - x_1 = \frac{\Delta X - x_1 \Delta Z}{Z_1} \qquad (6.14)$$

$$y_2 - y_1 = \frac{\Delta Y - y_1 \Delta Z}{Z_1} \qquad (6.15)$$

All the published methods for the recovery of the direction $(\Delta X/\Delta Z,\ \Delta Y/\Delta Z)$ are based on equations (6.14) and (6.15) (see [Ullman, 1979; Longuet-Higgins, 1981; Tsai and Huang, 1984]), which of course require the knowledge of the correspondence between points in the successive frames. In the sequel, we present a method for the recovery of the translational direction of a moving planar contour$(\Delta X/\Delta Z,\ \Delta Y/\Delta Z)$, without having to solve the point-to-point correspondence problem.

Consider again a coordinate system $OXYZ$ fixed with respect to the camera as in Figure 6.2 and let $A = \{(X_i, Y_i, Z_i)/i = 1, 2, 3, \ldots, n\}$, such that

$$Z_i = pX_i + qY_i + c, \qquad i = 1, 2, 3, \ldots, n$$

that is, the points are planar. Let the points translate rigidly with translation $(\Delta X,\ \Delta Y,\ \Delta Z)$, and let $\{(x_i, y_i)/i = 1, 2, 3, \ldots, n\}$ and $\{(x_i', y_i')/i = 1, 2, 3, \ldots, n\}$ be the projections of the set A before and after the translation, respectively.

Consider a point (x_i, y_i) in the first frame which has a corresponding one (x_i', y_i') in the second (dynamic) frame. For the moment we do not worry about where the point (x_i', y_i') is, but we do know that the following relations hold between these two points:

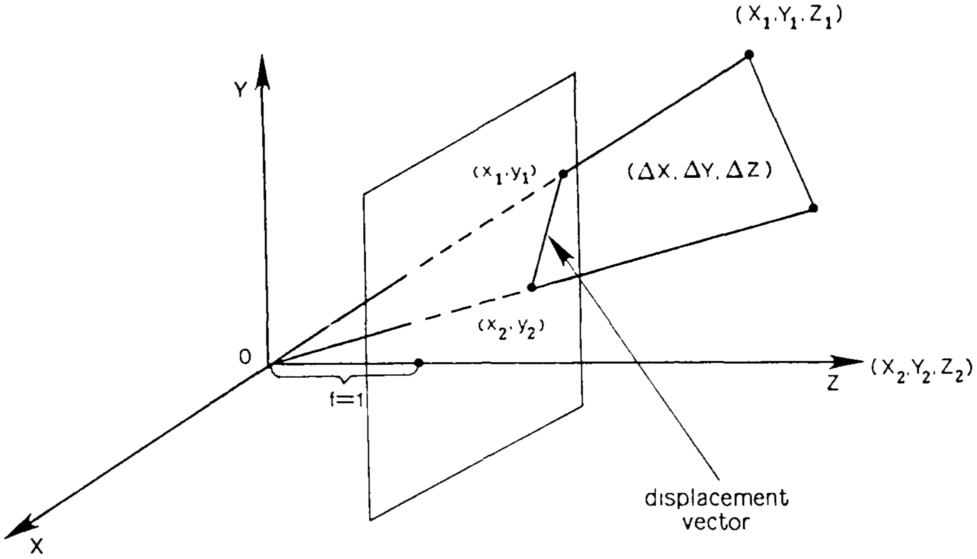


Figure 6.2: Motion of a point.

$$x_i' - x_i = \frac{f\Delta X - x_i \Delta Z}{Z_i} \tag{6.16}$$

$$y_i' - y_i = \frac{f\Delta Y - y_i \Delta Z}{Z_i} \tag{6.17}$$

where Z_i is the depth of the 3-D point whose projection (on the first dynamic frame) is the point (x_i, y_i). Taking now into account that

$$\frac{1}{Z_i} = \frac{f - px_i - qy_i}{cf} \tag{6.18}$$

the above equations become

$$x_i' - x_i = (f\Delta X - x_i \Delta Z)\frac{f - px_i - qy_i}{cf} \tag{6.19}$$

$$y_i' - y_i = (f\Delta Y - y_i \Delta Z)\frac{f - px_i - qy_i}{cf} \tag{6.20}$$

If we now write equation (6.19) for all the points in the two dynamic frames and sum the resulting equations up, we get

$$\sum_{i=1}^{n} (x_i' - x_i) = \sum_{i=1}^{n} \left[(f\Delta X - x_i \Delta Z)\frac{f - px_i - qy_i}{cf} \right]$$

or

$$\sum_{i=1}^{n} (x_i' - x_i) = \sum_{i=1}^{n} \left[\frac{f(f - px_i - qy_i)\Delta X - x_i\,(f - px_i - qy_i)\Delta Z}{c^{*}f} \right] \tag{6.21}$$

Similarly, if we do the same for equation (6.20), we get

$$\sum_{i=1}^{n} (y_i' - y_i) = \sum_{i=1}^{n} \left[(f\Delta Y - y_i \Delta Z)\frac{f - px_i - qy_i}{cf} \right]$$

or

$$\sum_{i=1}^{n} (y_i' - y_i) = \sum_{i=1}^{n} \left[\frac{f(f - px_i - qy_i)\Delta Y - y_i(f - px_i - qy_i)\Delta Z}{cf} \right] \tag{6.22}$$

At this point it has to be understood that equations (6.21) and (6.22) do not require our finding any correspondence.

By dividing equation (6.21) by equation (6.22), we get

$$\frac{\sum_{i=1}^{n} x_i' - \sum_{i=1}^{n} x_i}{\sum_{i=1}^{n} y_i' - \sum_{i=1}^{n} y_i} = \frac{\sum_{i=1}^{n} \left[\frac{\Delta X}{\Delta Z} f(f - px_{li} - qy_{li}) - (f - px_{li} - qy_{li})x_{li} \right]}{\sum_{i=1}^{n} \left[\frac{\Delta Y}{\Delta Z} f(f - px_{li} - qy_{li}) - (f - px_{li} - qy_{li})y_{li} \right]} \tag{6.23}$$

Equation (6.23) is a linear equation in the unknowns $\Delta X / \Delta Z$, $\Delta Y / \Delta Z$ and the coefficients consist of expressions involving summations of point coordinates in both dynamic frames; for the computation of the latter no establishment of any point correspondences is required.

So, if we consider a binocular observer, applying the above procedure in both left and right "eyes", we get two linear equations (of the form of equation (6.23)) in the two unknowns $\Delta X / \Delta Z$, $\Delta Y / \Delta Z$, which constitute a linear system that in general has a unique solution.

6.6.1. What the previous method is not about, an unexpected bonus and some problems

If one is not careful when analyzing the previous method, then he might think that all the method does is to correspond the center of mass of the image points before the motion with the center of mass of the image points after the motion, and then based on that retinal motion to recover three dimensional motion. But this is wrong, because perspective projection does not preserve simple ratios, and so the center of mass of the image points before the motion does not correspond to the center of mass of the image points after the motion. All the above method does is aggregation of the motion constraints; it does not correspond centers of mass.

It should be noted, however, that for the method to work the same number of points (contour periphery points) is required in both frames (before and after the motion). If the motion is not large, then this is true and no problem arises.

6.7. Detecting Unrestricted 3-D Motion

In the previous section we presented a method to recover the direction of the translation of a translating planar contour from the motion of its left and right images. It is clear that we did not use any depth information. In this section we present a method of recovering the motion parameters of a rigidly moving planar contour. Any rigid motion can be represented as a rotation around an axis that we can freely choose to pass through any point of our choice, plus a translation. The problem then is reduced to finding the translation and the rotation matrix.

So, suppose that we have four images of a moving planar contour (left and right before the motion, left and right after the motion). With the already presented methods, we can recover the orientation and depth of the 3-D contour before and after the motion; let (p_1, q_1, c_1) and p_2, q_2, c_2) be the parameters of the 3-D contour before and after the motion respectively. But since we know the contour in 3-D, we will do our analysis of the motion in 3-D, instead of the image plane.

So, let

$$C_1 = \{(X_i, Y_i, Z_i) | i = 1, \ldots, n\}$$

and

$$C_2 = \{(X_j', Y_j', Z_j') | j = 1, \ldots, m\}$$

be the two positions of the 3-D contour.

We assume that the rotation axis passes through the center of gravity of C_1. This has as an immediate consequence that the translation is given by the displacement of the center of the gravity between the two positions of the contour. So,

translation $= (\Delta X, \Delta Y, \Delta Z) = T =$ center of mass of C_2- center of mass of $C_1 =$

$$\left(\frac{\sum X_i'}{m} - \frac{\sum X_i}{n}, \frac{\sum Y_i'}{m} - \frac{\sum Y_i}{n}, \frac{\sum Z_i'}{m} - \frac{\sum Z_i}{n} \right) \quad (6.24)$$

It is obvious that we used different points in the two positions of the contour. Obviously, we did not need to do this. The methods that find the 3-D position of the contour do not address any "aperture in the large" problem. But the 3-D points are found from their projections and discretization effects may cause a small difference in the number of points of the two positions of the contour. We found that equation (6.24) gave good results as we will see in the section on experiments.

What remains to be found is the rotation matrix. But since we know the surface normals $n_1 = (p_1, q_1, -1)$, $n_2 = (p_2, q_2, -1)$ of the two positions of the contour, we can immediately find the rotation around an axis parallel to the plane of the contour C_1. Indeed, the angle θ between nq and n_2

$$\left(\cos\theta = \frac{n_1 \cdot n_2}{||n_1|| \, ||n_2||} \right)$$

gives the rotation angle around the axis

$$\frac{n_1 \times n_2}{||n_1 \times n_2||} = l.$$

The angle θ along with the axis l constitute a rotation matrix, R_1. It is obvious that R_1 is not the final rotation matrix because it misses rotation around an axis perpendicular to the world plane. In other words, if we apply to contour C_1 the rotation matrix R_1 and the translation T, then the result will not be contour C_2 but a contour C_1' which lies on the same plane as C_2, and has the same center of gravity of C_2. To find the missing rotation, we must find the angle that we have to rotate contour C_1' around an axis n which passes through the center of gravity of contour C_2 and is perpendicular to C_2.

To do that, we start rotating the contour C_1' until it coincides with contour C_2. This is done with small increments and the coincidence of the two contours (C_1' and C_2) is signaled by the maximization of their common area. The resulting angle Φ along with the axis n constitute a new rotation matrix R_2. Obviously, the final rotation matrix is given by $R = R_1 \cdot R_2$.

Finally, it is clear that the method described above will not work (rotation matrix R_2 will not be found) for some symmetric contours. If, for example, the 3-D contour is a circle, matrix R_2 cannot be found, since C_1' and C_2 coincide; or if the 3-D contour is a square and the rotation angle $\Phi = \pi/2$, then again matrix R_2 cannot be found. This simple fact is also obviously true for human observers who observe apparent motion and are asked to estimate the 3-D motion parameters. Figure 6.3 shows this scheme in detail.

6.8. Using a Monocular Observer

Extension of the above results can obviously be trivially generalized for a monocular observer who is translating with known motion.

We proceed now with the final section which describes experimental results based on the previous methods for the recovery of structure, depth and 3-D motion of a moving planar contour by a binocular or trinocular observer.

6.9. Stability Analysis

Here we present a theoretical stability analysis of some representative algorithms that were previously introduced. We analyze the behavior of the algorithm in Section 6.4 (correspondenceless stereo); the analysis of the stability of the rest of the algorithms is done in a similar way.

Recalling from Section 6.4.1, equation (6.10) is used for $k = 0$ in three different image zones for constructing a linear system with unknowns the parameters of the plane. This equation can finally be written as

$$p \left[\sum_{i=1}^{n} x_{li} \right] + q \left[\sum_{i=1}^{n} y_{li} \right] + rfn = \frac{1}{d} \left[\sum_{i=1}^{n} x_{li} - \sum_{i=1}^{n} x_{ri} \right]$$

or $C_1 p + C_2 q + rf = C_3$ where $C_1 = \frac{1}{n} \sum x_{li}$, $C_2 = \frac{1}{n} \sum y_{li}$ and $C_3 = \frac{1}{dn} \left[\sum_{i=1}^{n} x_{li} - \sum_{i=1}^{n} x_{ri} \right]$ and n the total number of points. The equation was recovered by assuming that the plane equations had the form $ps + qy + rZ = 1$ (because if we used the other form, then we would have to consider some special cases).

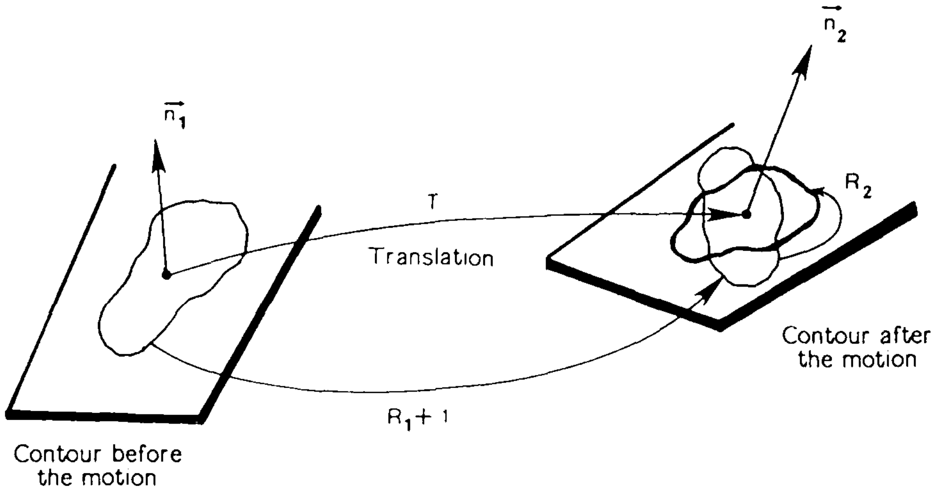


Figure 6.3: Contour before and after motion.

6.9.1. Error due to small perturbations of the points

The following analysis captures errors due to discretization and the uncertainty of the operators that extract interesting points. We will analyze the effect of random noise (in the image points) on the coefficients c_1, c_2 and c_3 and find out if consistent estimators arise. In that case, robustness is inferred. Suppose random noise $\epsilon_{x_{li}}$ is added to x_{li}, with $\epsilon_{x_{li}} \sim N(0,\sigma^2)$, i.e., $E(\epsilon_{x_{li}})=0$ and Var $(\epsilon_{x_{li}}=\sigma^2$. Similar noise is added to the y_{li} coordinates as well as to the coordinates in the right image frame. Also, assume that the stochastic variables corresponding to the noise added to different components are independent. Let x_{li}^o, y_{li}^o, etc. denote the observed x_{li}'s and y_{li}'s, etc., i.e., $x_{li}^o = x_{li} + \epsilon_{x_{li}}$, etc. Similarly, let c_i^o, $i=1, 2, 3$ be the observed coefficients. It is easy to check that $E(c_i^o)=c_i$ for $i=1, 2, 3$. On the other hand, checking the variances of the coefficients c_i^o, we have

$$\mathrm{Var}(c_1^o) = \frac{\sigma^2}{n} \to o \quad \text{as } n \to \infty$$

$$\mathrm{Var}(c_2^o) = \frac{\sigma^2}{n} \to o \text{ as } n \to \infty$$

and

$$\mathrm{Var}(c_3^o) = \frac{2\sigma^2}{d^2 n} \to o \text{ as } n \to \infty.$$

So, when random noise is added to the image points, the coefficients c_i^o, $i=1, 2, 3$ are consistent estimators of the corresponding true coefficients.

6.9.2. Error due to the addition and deletion of random noise

Here we analyze the effect of drop-ins and drop-outs. More specifically, there will be points in the left image frame whose corresponding ones won't be found by the interest point operator in the right image frame, and vice versa. Because this effect is very hard to capture quantitatively and it will depend on the particular "point extraction" operator, we do our analysis by assuming that this problem is realized by adding random points to the different frames. Let's analyze the effect of adding m_1 and m_2 noise points to the left and right images respectively. We assume that these points are distributed randomly (uniformly) over the image frames, i.e., if (x_{li}^N, y_{li}^N) denotes the i^{th} noise point in the left frame, then

$$x_{li}^N \sim \mathrm{U}(-k,k), \quad y_{li}^N \sim \mathrm{U}(-k,k)$$

and they are independent, where the image is represented in the box (Fig. 6.3.1).

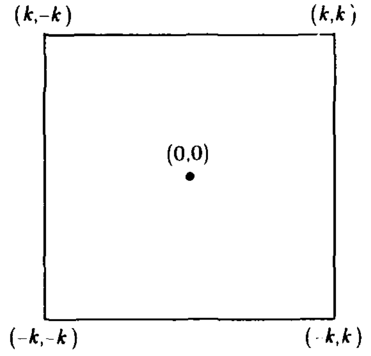


Figure 6.3.1.

The same symbolism is assumed for the noise points in the right image frame and the noise is assumed to be independent. Let c_1^*, c_2^* and c_3^* be estimators of the coefficients that were defined in the previous sections after adding noise points. We further assume (for simplicity of the calculations and without loss of generality) that $m_1 = m_2 = m$. Then we have

$$c_1^* = \frac{1}{n+m}\left(\sum_{i=1}^{n} x_{li} + \sum_{i=1}^{m} x_{li}^N\right)$$

$$c_2^* = \frac{1}{n+m}\left(y_{li} + \sum_{i=1}^{m} y_{li}^N\right)$$

$$c_3^* = \frac{1}{d(n+m)}\left[\sum_{i=1}^{n} x_{li} \; \sum_{i=1}^{n} x_{ri} + \sum_{i=1}^{m} x_{li}^N \; \sum_{i=1}^{m} x_{ri}^N\right]$$

Since the mean of the noise points is assumed to be zero, i.e., $E(x_{li}^N) = 0$, etc., we have

$$E(c_1^*) = \frac{n}{n+m}c_1,\ E(c_2^*) = \frac{n}{n+m}c_2$$

$$\text{and } E(c_3^*) = \frac{n}{n+m}c_3.$$

So, $E(c_i^*) = \frac{n}{n+m}c_i,\ i = 1, 2, 3.$

Thus, unlike the perturbation noise, throwing in random points makes the estimates c_i^* underestimate c_i by a factor of $\frac{n}{n+m}$ for $i = 1, 2, 3$.

Let us now examine the consistence of the estimates. We have

$$\text{Var}(x_{li}^N) = \frac{1}{2k}\int_{-k}^{k} x^2 dx = \frac{1}{6k}x^3 \int_{-k}^{k} = \frac{k^2}{3}$$

So,

$$\text{Var}(c_3^*) = \frac{1}{m+n}2m\frac{k^2}{3} = \frac{2m}{3(m+n)}k^2$$

This is enough to convince us that, unlike random perturbation of points, throwing in points at random makes the estimators based on the image points inconsistent. However, from actual experiments we observed that the effect of this kind of error hardly affected the results, when the difference in the number of points in the left and right image frame was about 20% of the total number of points.

6.9.3. Error due to numerical instabilities

The fact that we are left with the soiution of a linear system for our correspondenceless stereo does not mean that the solution will be robust. The matrix might be ill-conditioned or critically ill-conditioned. The analysis is the same as in Section 5.5 and will be omitted here. Experiments showed robustness but a theoretical classification of pathological cases (point distributions that create critical ill-conditioning) is still needed.

6.9.4. General stability comments

We think that the level of robustness or stability is an essential part of the Marr paradigm (that Marr left implicit in his writings). In this paper we used some simple statistical analysis to demonstrate the behavior of our algorithms in the presence of noise. One can resort to worst case or average error analysis, assuming some probability distribution of the input. A nice methodology for computing the effects of discretization error is presented in [Kamgar-Parsi and Kamgar-Parsi, 1987] and can be applied here as well. We think that research on robustness should go even further. Some problems (such as structure from motion) are quite unstable (as observed) for all algorithms that have been proposed. Maybe, then, the problem itself is unstable by its nature and should be formulated differently. For this to happen, one should have to prove that the problem is unstable, regardless of the algorithm used. In a sense, one would have to consider the space of all problems, define a probability distribution in the space of all problems and then compute the probability distribution of the chance a problem has to be unstable, where instability is expressed as the distance from a set of ill-posed problems. Such an approach has been initiated in numerical analysis [Demmel, 1987].

6.10. Experiments

We experimented with synthetic and natural images. Again the acquisition of natural images was done through a solid state Panasonic camera and the motion control through a Merlin American Robot arm.

6.10.1. Synthetic images

The following experiments implement the algorithms of Section 6.4.

Figure 6.4 shows the projections of a set of planar points on both the left and right frames. The frame on top is the superposition of the left and right frames. The actual parameters of the plane were $p = 0.0$, $q = 0.0$, $c = 10000$, while the number of points was equal to 1000. We did not include any noise in our pictures. The computed ones were $\mathbf{P} = -0.0$, $\mathbf{Q} = -0.0$, $C = 10000.0$.

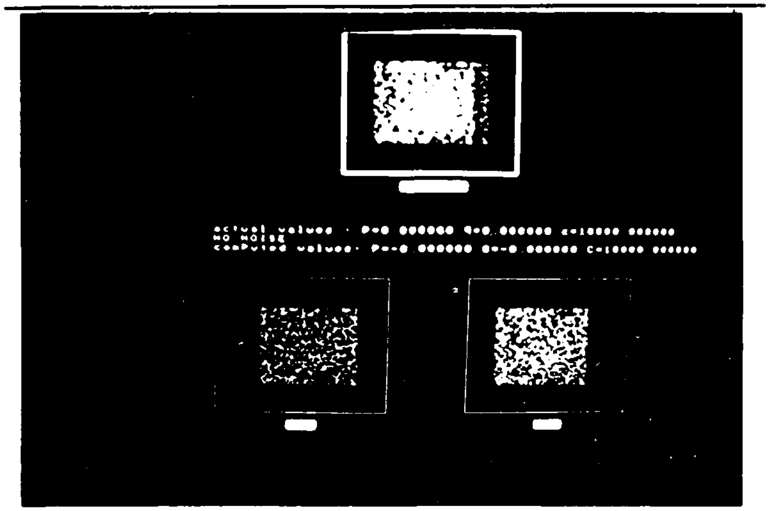
Figure 6.5 shows the projections of a set of planar points on both the left and right frames. The frame on top is the superposition of the left and right frames. The actual parameters of the plane were: $p = 1.0$, $q = 1.0$, $c = 10000$, while the number of points was equal to 1000. We did not include any noise in our pictures. The computed ones were $\mathbf{P} = 0.98$, $\mathbf{Q} = 1.00$, $C = 9809.8$.

Figure 6.6 shows the projections of a set of planar points on both the left and right frames. The frame on top is the superposition of the left and right frames. The actual parameters of the plane were $p = 1.0$, $q = 1.0$, $c = 10000$, while the number of points was equal to 1000. We included 5% noise in the left frame and 7% in the right one. By noise here we mean putting points at random in order to capture the effect of the imperfectness of *interesting point extraction operations*. In actual real images, we would first have to extract interesting points. The computed ones were $\mathbf{P} = 1.7$, $\mathbf{Q} = 1.2$, $C = 10266.7$.

The next experiments test the algorithms in Section 6.6 for determining direction of translation for a set of planar points as well as for a planar contour, and the algorithms for determining unrestricted 3-D motion.

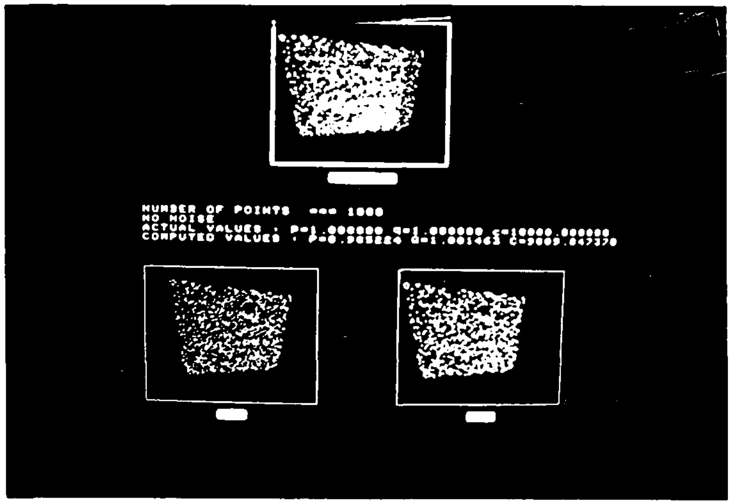
Figure 6.7, 6.8 and 6.9 implement the algorithm of equations (6.23). The frames at the bottom represent the projections of a set of 3D planar points on the left and right eyes respectively. The two frames at the top represent the projections of the same set of points, after it has been translated. The actual direction of translation, the computed one, the number of points and the noise are seen in the respective figures. The rest of the figures show experimental results for computation of shape, direction of translation and unrestricted 3D motion from the algorithms in Sections 6.5, 6.6 and 6.7

Figure 6.10 shows the images of a translating planar contour (human figure) taken by a binocular system at two different time instants. The actual orientation of the contour in space was $(p,q) = (10,5)$ and the actual direction of translation $(dx/dz,\ dy/dz) = (-4,6)$. Our program recovered orientation $(p,q) = (10.00007,\ 5.000297)$ and direction of translation $(dx/dz,\ dy/dz) = (\ 4.000309,\ 6.00463)$. Figure 6.11 shows again the perspective images of a translating planar contour taken by a binocular system at two different time instants. The actual orientation of the contour was $(p,q) = (-25,30)$ and the direction of translation $(dx/dz,\ dy/dz) = (50,60)$. The computed orientation from these images was (p,q) $(-24.99,\ 30.000021)$ and the computed



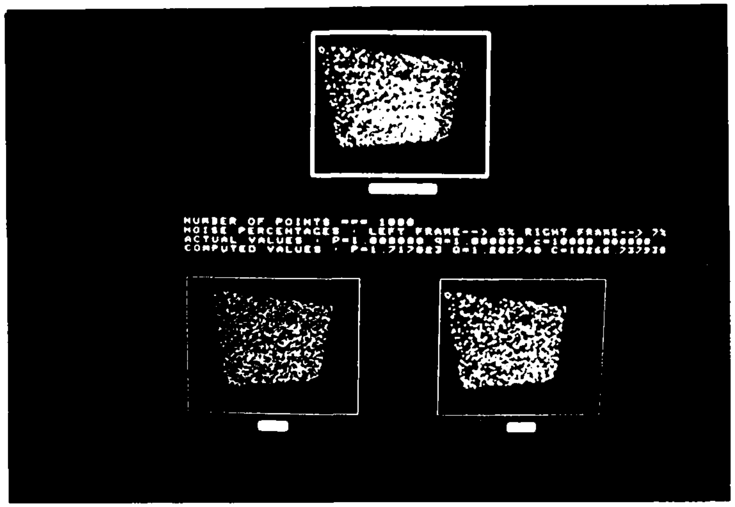
```
actual values : P=0.000000 Q=0.000000 C=10000.000000
NO NOISE
computed values: P=-0.000000 Q=-0.000000 C=10000.000000
```

Figure 6.4: First experiment.



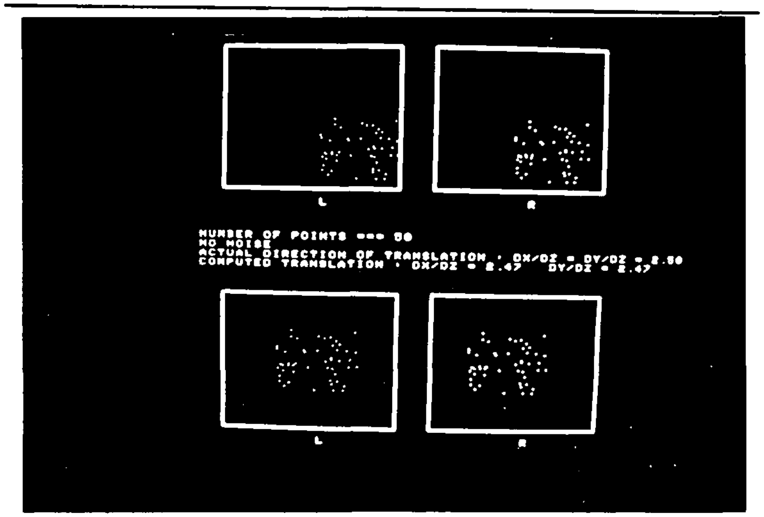
```
NUMBER OF POINTS --- 1000
NO NOISE
ACTUAL VALUES : P=1.000000 Q=1.000000 C=10000.000000
COMPUTED VALUES : P=0.985224 Q=1.001463 C=9809.847370
```

Figure 6.5: Second experiment.



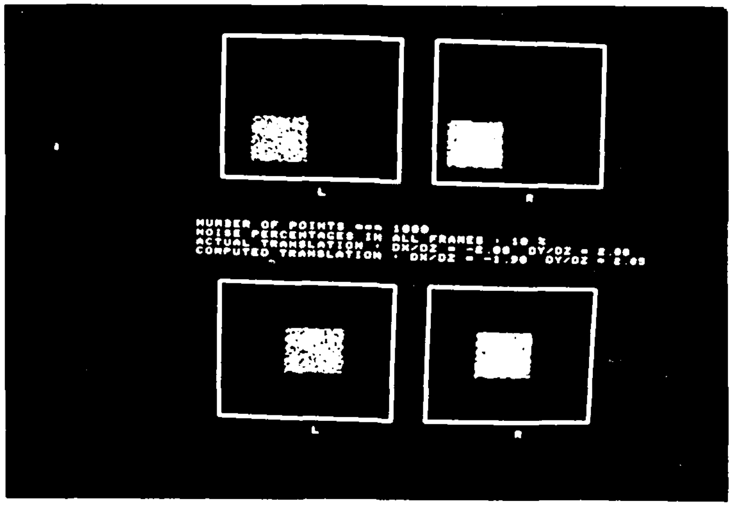
```
NUMBER OF POINTS --- 1000
NOISE PERCENTAGES : LEFT FRAME--> 5% RIGHT FRAME--> 7%
ACTUAL VALUES : P=1.000000 Q=1.000000 C=10000.000000
COMPUTED VALUES : P=1.717023 Q=1.202740 C=10266.737930
```

Figure 6.6: Third experiment.



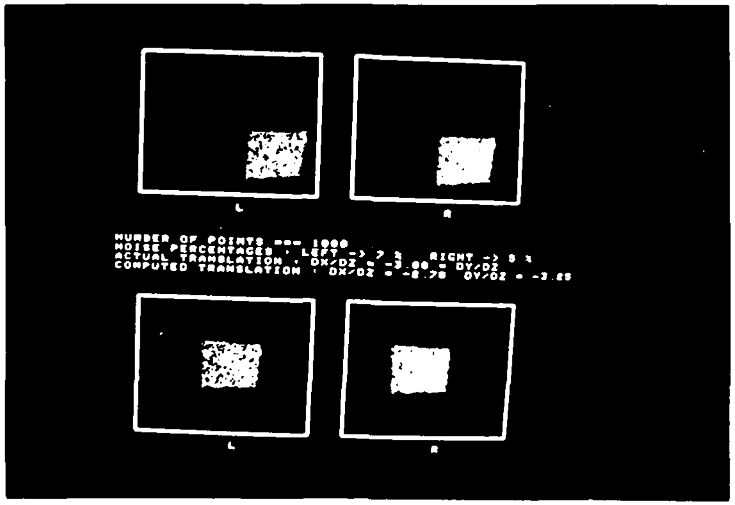
```
NUMBER OF POINTS --- 50
NO NOISE
ACTUAL DIRECTION OF TRANSLATION : DX/DZ = DY/DZ = 2.50
COMPUTED TRANSLATION : DX/DZ = 2.47   DY/DZ = 2.47
```

Figure 6.7: Fourth experiment.



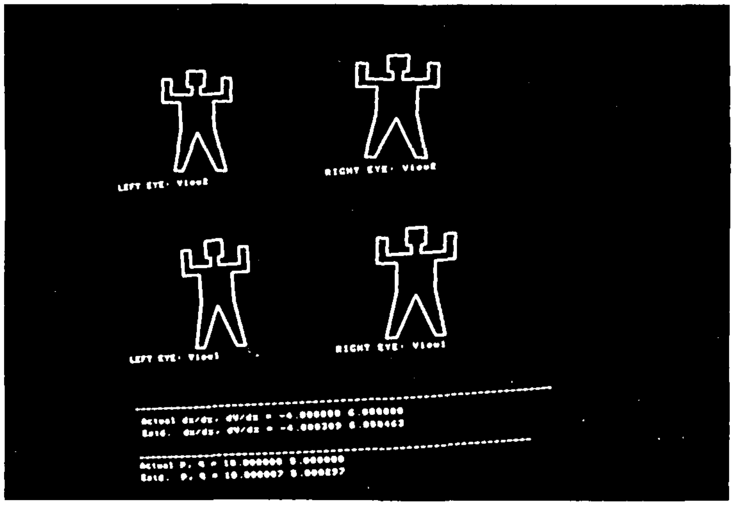
```
NUMBER OF POINTS ==> 1000
NOISE PERCENTAGES IN ALL FRAMES : 10 %
ACTUAL TRANSLATION : DX/DZ = -2.00  DY/DZ = 2.00
COMPUTED TRANSLATION : DX/DZ = -1.90  DY/DZ = 2.05
```

Figure 6.8: Fifth experiment.



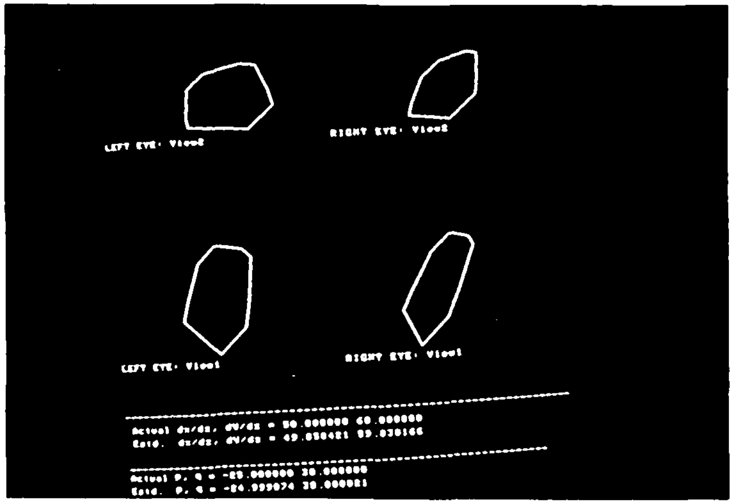
```
NUMBER OF POINTS ==> 1000
NOISE PERCENTAGES : LEFT -> 7 %   RIGHT -> 5 %
ACTUAL TRANSLATION : DX/DZ = -3.00 = DY/DZ
COMPUTED TRANSLATION : DX/DZ = -2.70  DY/DZ = -3.25
```

Figure 6.9: Sixth experiment.



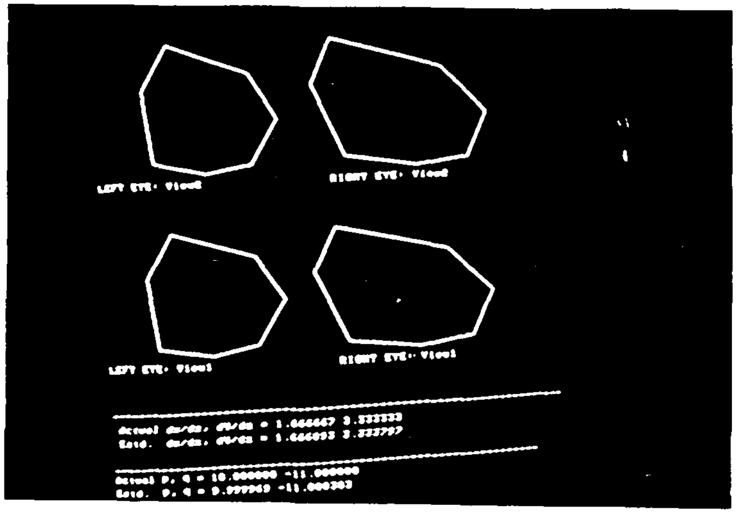
```
Actual dU/dz, dV/dz = -4.000000 6.000000
Estd.  dU/dz, dV/dz = -4.000309 6.000463

Actual P, Q = 10.000000 5.000000
Estd.  P, Q = 10.000007 5.000297
```

Figure 6.10: Seventh experiment.



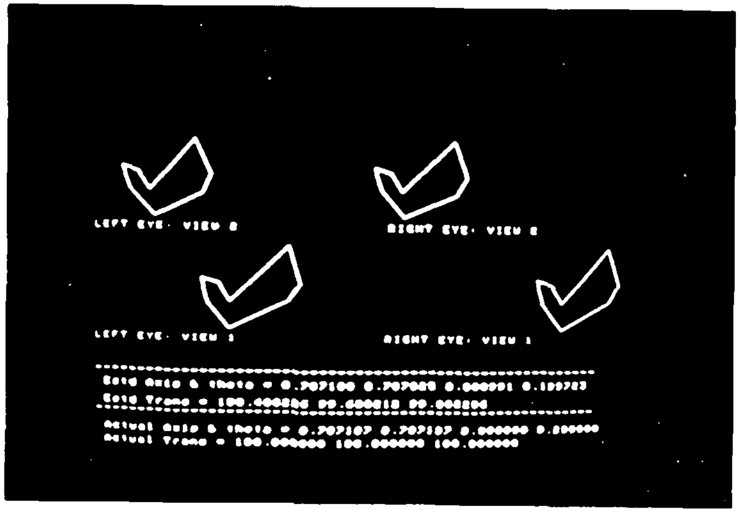
```
Actual dU/dz, dV/dz = 50.000000 60.000000
Estd.  dU/dz, dV/dz = 49.050421 59.030166

Actual P, Q = -25.000000 30.000000
Estd.  P, Q = -24.999974 30.000081
```

Figure 6.11: Eighth experiment.



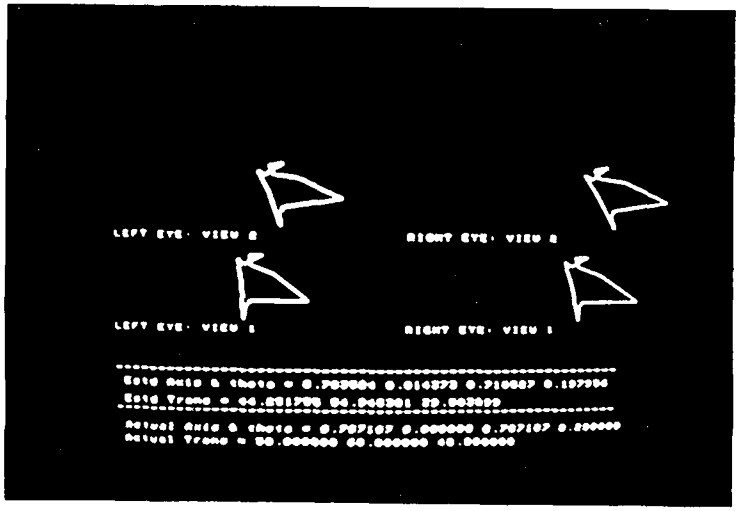
```
Actual du/dz, dv/dz = 1.666667 3.333333
Estd.  du/dz, dv/dz = 1.666093 3.333797

Actual P, Q = 10.000000 -11.000000
Estd.  P, Q = 9.999969 -11.000303
```

Figure 6.12: Ninth experiment.



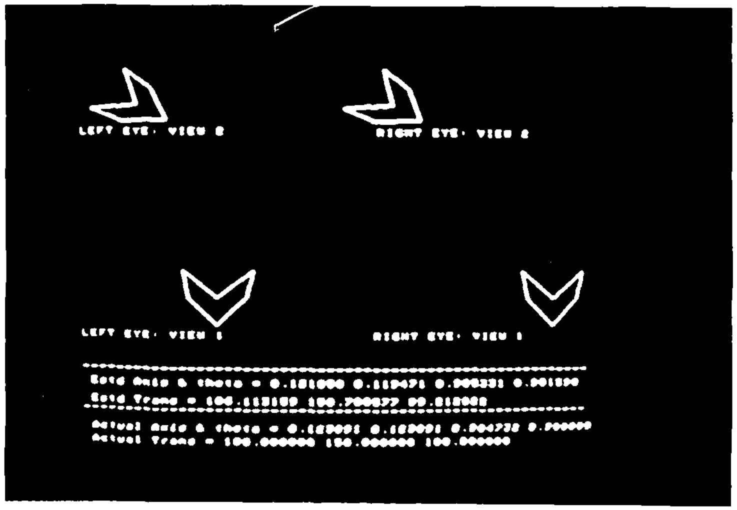
```
---------------------------------------------------------------
Estd Axis & theta = 0.707100 0.707086 0.000901 0.199723
Estd Trans = 100.000286 99.000018 99.008206
---------------------------------------------------------------
Actual Axis & theta = 0.707107 0.707107 0.000000 0.200000
Actual Trans = 100.000000 100.000000 100.000000
```

Figure 6.13: Tenth experiment.



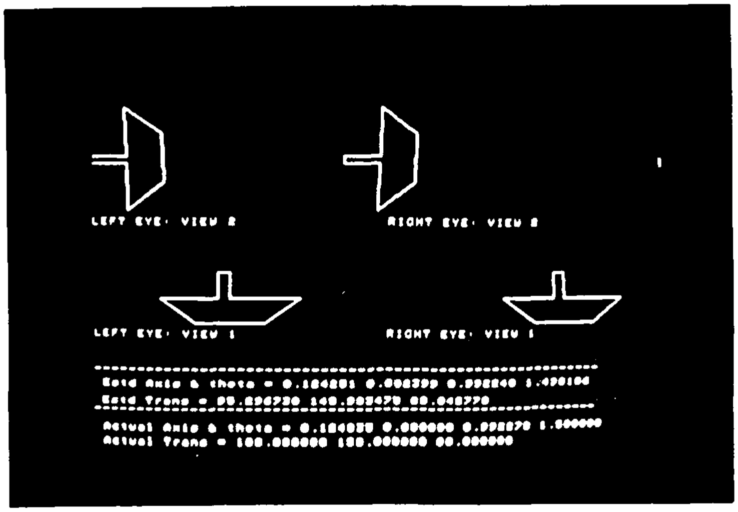
```
---------------------------------------------------------------
Estd Axis & theta = 0.703004 0.014373 0.710087 0.197396
Estd Trans = 64.881798 84.968301 39.003099
---------------------------------------------------------------
Actual Axis & theta = 0.707107 0.000000 0.707107 0.200000
Actual Trans = 50.000000 60.000000 40.000000
```

Figure 6.14: Eleventh experiment.



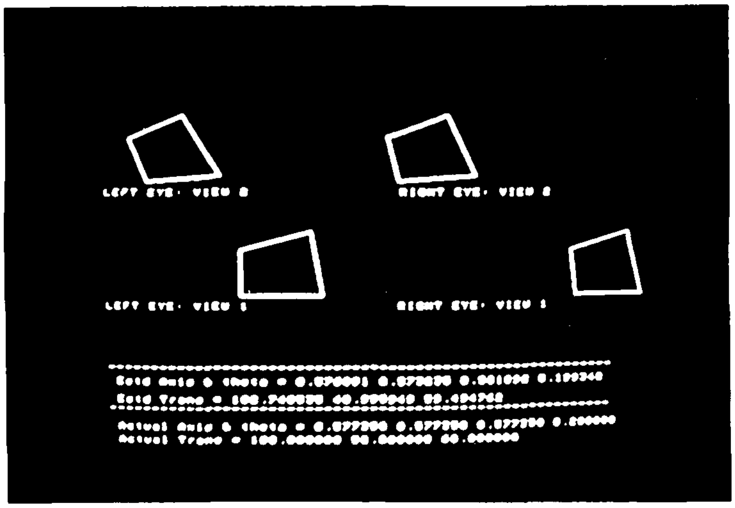
```
---------------------------------------------------------------
Estd Axis & theta = 0.181000 0.119471 0.990331 0.901190
Estd Trans = 100.113159 100.700077 99.812088
---------------------------------------------------------------
Actual Axis & theta = 0.120091 0.120091 0.904738 0.900000
Actual Trans = 100.000000 100.000000 100.000000
```

Figure 6.15: Twelfth experiment.



```
Estd Axis & theta = 0.124201 0.002399 0.992840 1.490106
Estd Trans = 95.296730 145.985470 80.040770
Actual Axis & theta = 0.124035 0.000000 0.992278 1.500000
Actual Trans = 100.000000 150.000000 80.000000
```

Figure 6.16: Thirteenth experiment.



```
Estd Axis & theta = 0.576901 0.573670 0.581696 0.199340
Estd Trans = 102.740870 49.000840 59.494762
Actual Axis & theta = 0.577350 0.577350 0.577350 0.200000
Actual Trans = 100.000000 50.000000 60.000000
```

Figure 6.17: Fourteenth experiment.

direction of translation $(dx/dz,\ dy/dz)$ = (49.858421, 59.830266). Figure 6.12 shows the perspective images of a translating planar contour taken by a binocular system at two different times. The actual orientation of the contour was $(p,q) = (10,-11)$ and the direction of translation $(dx/dz, dy/dz) = (1.66, 3.33)$. The estimated parameters from these images were $(p,q) = (9.99,\ -11.000383)$ and $(dx/dz, dy/dz) = (1.66, 3.33)$.

The experiments to determine the general motion parameters are shown in Figures 6.13-6.17. The actual and computed parameters are recalculated with respect to the coordinate system of the left camera. In Figure 6.13 the actual translation was (100, 100, 100) and actual rotation was 0.2 radians around the axis (0.707, 0.707, 0); the estimated values were translation = (100.4, 99.6, 99.8) and rotation = 0.1997 radians around the axis (0.707, 0.707, 0). The results for the next figure were as follows: actual translation (50, 60, 40) and actual rotation = 0.2 radians around the axis (0.707, 0, 0.707). The estimated translation was (44.25, 54.94, 39.53) and the estimated rotation was 0.1980 radians around the axis (0.704, 0.014, 0.711). Figure 6.15 shows the actual translation as (100, 150, 100) and rotation Of 0.9 radians around the axis (0.123, 0.123, 0.985); the estimates were translation = (106.11, 150.7, 99.21) and rotation = 0.902 radians around the axis (0.121, 0.119, 0.985). The ship in Figure 6.16 was translated by (100, 150, 80) and rotated by 1.5 radians around the axis (0.124, 0, 0.992); the recovered parameters were translation = (95.30, 145.98, 80.04) and rotation = 1.49 radians around the axis (0.124, 0, 0.992). Figure 6.17 shows the actual parameters as translation = (100, 50, 60) and rotation = 0.2 radians around the axis (0.577, 0.577, 0.577). The estimated parameters were translation = (102.75, 49, 59.49) and rotation = 0.199 radians around the axis (0.577, 0.573, 0.582).

Note: *All the parameters involved in the above experiments that have a dimension of length* (L^1 M^0 T^0) *are calculated in pixels, where 1 pixel = 100 μm.*

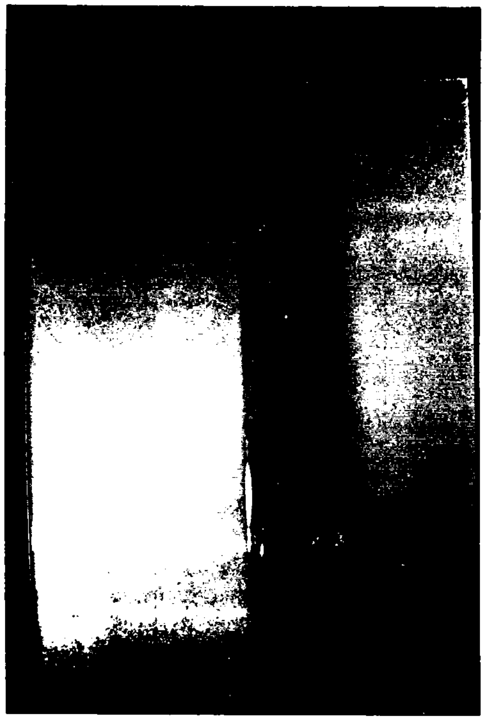


Figure 6.18: Block (left image) before rotation.

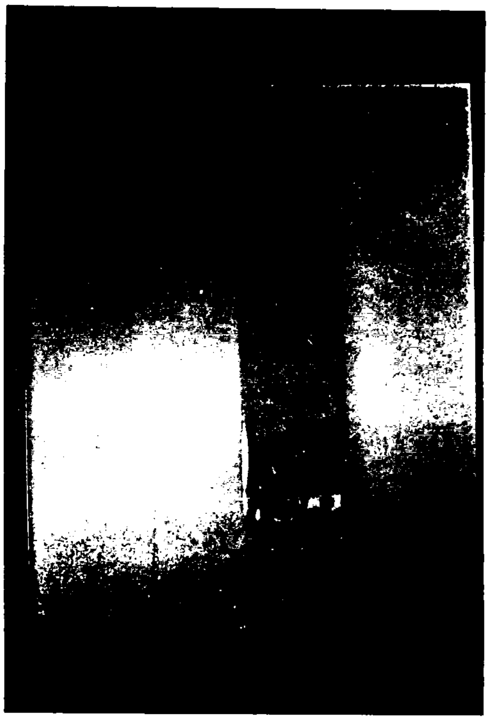


Figure 6.19: Block (right image) before rotation.

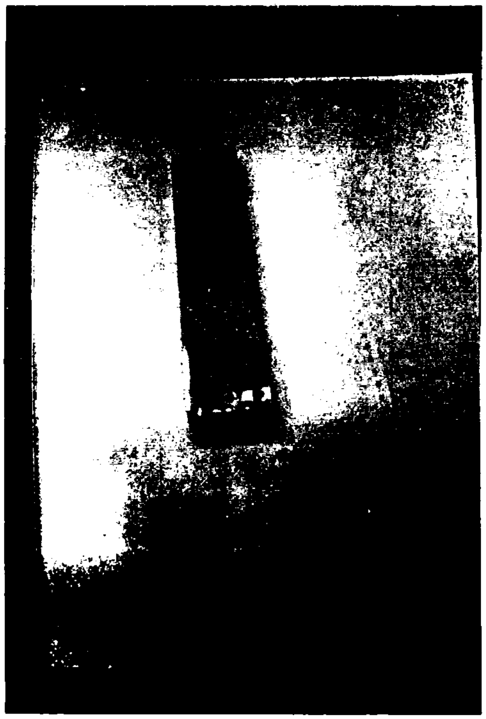


Figure 6.20: Block (left image) after rotation.

Figure 6.21: Block (right image) after rotation.

6.10.2. Natural images

Here we describe an experiment where we used the algorithm of Section 6.4 to compute the orientation of a planar block. We measure again the change in shape, not the actual shape, in order to avoid ambiguities in ground truth measurement. Figures 6.18 and 6.19 show the left and right images of a block. Figures 6.20 and 6.21 show again the left and right images of the block after the camera has considerably rotated in order to change the orientation of the block. We measure the two surface normals, before and after the rotation, and then we compute the angle between the normals, whose ground truth we know from the rotation of the camera. In this experiment the ground truth for the angle between the surface normals was $\theta = 30°$ and the estimated one was $\theta = 33.103°$. We also carried out experiments for the computation of motion (only direction of translation because unrestricted 3-D motion requires computation of depth whose ground truth we couldn't estimate). The accuracy here was better than 5 percent in all cases.

7. COMBINING INFORMATION FROM OTHER CUES

We talked about combining some cues to achieve unique and robust solutions to the computation of some intrinsic images. Some researchers have combined cues in the past to compute 3-D properties, such as Waxman and Duncan (binocular image flows: stereo and retinal motion), Richards (stereo and motion) and Grimson (shading and stereo). The analysis here is by no means complete. More cues such as color, discontinuities, nonplanar contours, can and should be combined with other sources of information. We hope that this work will not only serve as a source of technical results but also as a general methodology in computer vision problems. By trying to solve problems in a correspondenceless way we do not mean that correspondence is useless. On the contrary, correspondence is a very basic process [Ullman, 1979] but it is a very hard problem to solve. We also hope that this research might generate some research on attacking the hard correspondence problem that is the basis of many visual processes.

8. CONCLUSIONS AND FUTURE DIRECTIONS

In this paper we claim that low-level vision computations should be done in such a way that uniqueness and robustness of the computations is guaranteed and that visual computations can be done in this way. We justified our claims by examining several problems, such as shape from texture, shape from shading, visual motion analysis, shape and motion from contour and some cases of stereo.

The problem of understanding vision and building intelligent machines with a visual sense is very hard and by no means solved. We have argued that a very large part of today's research is analyzing visual capabilities, i.e. research is concentrating on topics that correspond to identifiable modules in the human visual system. And even though it is not at all clear what the topics are that correspond to identifiable modules in the human visual system, research has shown that shading, texture, motion, contours and stereo are cues that help to understand the extrapersonal space.

There is no doubt that vision is full of redundancy and there is a lot of information in the image which if used correctly will give rise to constraints which will guarantee uniqueness and robustness of the visual computations. We have demonstrated this for the case of the problems that appeared in Sections 4, 5 and 6. Obviously, we need to obtain robust and unique visual computations if we ever want to advance our understanding of vision.

There is a standard way to design large and complex information systems as research in computational fields has shown [Feldman, 1985].

(1) First we divide the system into functional components which break the overall task into autonomous parts, and analyze these components (Fig 2.3).

(2) Then we must choose the representation of information within the subsystems and the language of communication among them.

(3) After this, the details of the systems are tested individually, in pairs and all together.

In this paper, in the course of our analysis of a visual system (machine or biological), we started with the first two steps and a part of the third, and we did this for some subsystems (texture, shading, motion, contours, stereo). Our results can be summarized best in Figure 3.2.

There are more subsystems to be analyzed such as color, nonplanar contours, recognition of objects, navigation modules, and many others. The analysis of all of these constitutes our future research, as well as the research of several others. More importantly, our immediate future research will be devoted to the third step, where we have to test the subsystems all together. Finally, the robustness of the introduced algorithms should be established in a unified theoretical way. In the present work, our robustness justification was largely based on the fact that in most of the cases redundancy did the job. This paper, besides the technical results, has attempted to establish a paradigm in which vision problems should be addressed. Always keeping in mind the Marr paradigm [1981], we proposed that information should be combined, from different sources when possible, in order to achieve uniqueness and robustness of visual computations and at the same time making the minimal number of assumptions. The mathematical vocabulary for combining information from different sources, which is deterministic, is that of discontinuous regularization, one of our current research topics.

REFERENCES

Aloimonos, J., "Computing intrinsic images", Ph.D. thesis, University of Rochester, Rochester, NY.

Aloimonos, J. and M. Spetsakis, "A unified error analysis for visual algorithms", Technical Report 1798, Center for Automation Research, University of Maryland, College Park, 1987.

Aloimonos, J. and J. Hervé, "Correspondenceless detection of depth and motion for planar surfaces", technical report in preparation, Center for Automation Research, University of Maryland, College Park, 1987

Aloimonos, J., "Detection of surface orientation from texture", *Proc. CVPR*, 1986.

Amari, S., Personal communication, 1986.

Ballard, D.H., "Parameter networks", *Artificial Intelligence* **22**, 235-267, 1984.

Brady, M. and A. Yuille, "An extremum principle for shape from contour", *IEEE Trans. PAMI* **6**, 288-301, 1984.

Brooks, M. and B.K.P. Horn, "Shape and source from shading", *Proc. IJCAI* 1985, Los Angeles, CA.

Brooks, M., Ph.D. Thesis, University of Sussex, England, 1984.

Brown, C.M., D.H. Ballard, and E. Rainero, "Constraint propagation in shape from shading", *Proc. Image Understanding Workshop*, 1983.

Demmel, J., "The probability that a numerical analysis problem is unstable", Technical Report, Courant Institute of Math. Sciences, 19°7.

Dunn, S.M., L.S. Davis and L. Janos, "Efficient recovery of shape from texture", *IEEE Trans. PAMI* **5**, 485-492, 1983.

Feldman, J.A. "Four frames suffice: A provisional model of vision and space", *Behavioral and Brain Sciences,* June 1985.

Grimson, W.E.L., "Binocular shading and visual surface reconstruction", *CVGIP* **28**, 19-44, 1984.

Hildreth, E.C., "Computations underlying the measurement of visual motion", *Artificial Intelligence* **23**, 309-354, 1984.

Horn, B.K.P., "Obtaining shape from shading information", in *Psychology of Computer Vision*, P.H. Winston, (ed.), McGraw-Hill, New York, 115-155, 1975.

Horn, B.K.P., *Robot Vision,* McGraw-Hill, 1986.

Horn, B.K.P., "Understanding image intensities", *Artificial Intelligence* **8**, 201-231, 1977.

Horn, B.K.P. and B.G. Schunck, "Determining optical flow", *Artificial Intelligence* **17**, 185-204, 1981.

Horn, B.K.P. and R.W. Sjoberg, "Calculating the reflectance map", *Applied Optics* **18**, 1770-1779, 1979.

Huang, T.S. and R.Y. Tsai, "Image sequence analysis: Motion estimation", in *Image Sequence Analysis*, T.S. Huang (ed.), Springer-Verlag, 1981.

Huang, T.S. and S.D. Blostein, "Robust algorithms for motion estimation based on two sequential stereo image pairs", *Proc. CVPR*, 518-523, 1985.

Hummel, R., "Image reconstruction from zero-crossings and gradients", personal communication, 1986.

Ikeuchi, K. and B.K.P. Horn, "Numerical shape from shading and occluding boundaries", *Artificial Intelligence* **17**, 141-184, 1981.

Ito, E. and J. Aloimonos, "Recovering transformation parameters from images: Theory", *Proc. IEEE Conference on Robotics and Automation*, 1987.

Kamgar-Parsi, B. and Kamgar-Parsi, B., "Evaluation of quantization error in computer vision", CAR-TR-316, Center for Automation Research, University of Maryland, College Park, January 1987.

Kanade, T., "Recovery of three-dimensional shape of an object from a single view", CMU-CS-79-A53, Department of Computer Science, Carnegie-Mellon University, 1979.

Kanatani, K., "Detection of surface orientation and motion from texture by a stereological technique", *Artificial Intelligence* **23**, 213-237, 1984.

Kanatani, K., "Structure from motion without correspondence: general principle", *Proc. DARPA Image Understanding Workshop*, Miami Beach, FL, December 1985, 107-116.

Lee, D. and T. Pavlidis, "A linear theory of discontinuous regularization", Personal communication, 1987.

Lipschutz,M., "Differential Geometry", *Schaum's Outline Series,* 1969.

Longuet-Higgins, H.C. and K. Prazdny, "The interpretation of a moving retinal image", *Proc. Royal Soc. London* **B 208**, 385-397, 1984.

Longuet-Higgins, H.D., "A computer algorithm for reconstructing a scene from two projections", *Nature* **293**, 133-135, 1981.

Marr, D., *Vision,* W.H. Freeman, San Francisco, 1981.

Marr, D. and E. Hildreth, "Theory of edge detection", *Proc. Royal Soc. London* **B 207**, 187-217, 1980.

Marr, D. and T. Poggio, "A theory of human stereo vision", *Proc. Royal Soc. London* **B 207**, 301-328, 1979.

Marr, D. and T. Poggio, "From understanding computation to understanding neural circuitry", in *Neural Mechanisms in Visual Perception*, Neuroscience Research Program Bulletin, Poppel, E. et al. (eds.), **15**, 470-488, 1977.

Milenkovich, V. and T. Kanade, "Trinocular stereo", *Proc. Image Understanding Workshop*, Miami, FL, 1985.

Morozov, V.A., *Methods for Solving Incorrectly Posed Problems*, Springer, 1984.

Nagel, H.H., "Displacement vectors derived from second order intensity variations in image sequences", *CVGIP* **21**, 85-117, 1983.

Negahdaripour, S. and B.K.P. Horn, "Direct passive navigation", *Proc. Image Understanding Workshop*, Miami, FL, 1985.

Ohta, Y., K. Maenobu, and T. Sakai, "Obtaining surface orientation from texels under perspective projection", *Proc. 7th IJCAI*, Vancouver, Canada, 746–751, 1981.

Pentland, A.P., "Shading into texture", *Proc. AAAI*, 269–273, 1984.

Pentland, A.P., "Local shading analysis", *IEEE Trans. PAMI* **6**, 170–187, 1984.

Pentland, A.P., "Finding the illuminant direction", *J. Optical Society of America* **72**, 448–455, 1982.

Poggio, T. and C. Koch, "Ill-posed problems: from computational theory to analog networks", *Proc. Royal Soc. London* **B**, 1985.

Prazdny, K., "Determining the instantaneous direction of motion from optical flow generated by a curvilinearly moving observer", *CVGIP* **17**, 238-248, 1981.

Prazdny, K., "Egomotion and relative depth map from optical flow", *Biol. Cybernetics* **26**, 87–102, 1980.

Richards, W., "Structure from stereo and motion", AI Memo 731, MIT AI Lab, Cambridge, MA. Also in *J. Optical Society of America* **A2**, 343-349, 1985.

Shulman, D. and J. Aloimonos, "A theory of regularization preserving discontinuities", technical report in preparation, Center for Automation Research, University of Maryland, College Park, 1987.

Shafer, S., "Shape from shading under perspective projection", personal communication.

Stevens, K.A., "Surface perception from local analysis of texture and contour", Ph.D. thesis, Technical Report TR 512, MIT, Cambridge, MA, 1979.

Strat, T.M., "A numerical method for shape and shading from a single image", M.S. thesis, Department of Electrical Engineering and Computer Science, MIT, 1979.

Terzopoulos, D., "Regularization of inverse problems involving discontinuities", *IEEE Trans. PAMI* **8**, 413–425, 1986.

Tichonov, A.N. and V.Y. Arsenin, *Solution of Ill-Posed Problems*, Winston and Wiley, Washington, DC, 1977.

Tsai, R.Y. and T.S. Huang, "Uniqueness and estimation of three dimensional motion parameters of rigid objects", in *Image Understanding 1984*, S. Ullman and W. Richards (eds.), New Jersey: Ablex Publishing Co., 1984.

Ullman, S., *The Interpretation of Visual Motion*, MIT Press, Cambridge, MA, 1979.

Waxman, A. and J. Duncan, "Binocular image flows", *Proc. Workshop on Motion*, Charleston, SC, 1986.

Webb, J.A. and J.K. Aggarwal, "Shape and correspondence", *CVGIP* **21**, 145 160, 1983.

Witkin, A., "Recovering surface shape and orientation from texture", *Artificial Intelligence* **17**, 17–45, 1981.

Witkin, A., "Shape from contour", Ph.D. thesis, Department of Psychology, MIT, 1980.

APPENDIX A

A.1. Geometric Correspondence Between Points in the Scene and the Image

A.1.1. Perspective projection

Consider an ideal pinhole at a fixed distance in front of an image plane (see Figure A.1). Let us assume that an enclosure is provided so that only light coming through the pinhole can reach the image plane. Given that light travels along straight lines, each point in the image corresponds to a particular direction defined by a ray from that point through the pinhole. This is what we know as perspective projection. In the sequel, in order to simplify the resulting equations, we consider the nodal point of the eye (pinhole) behind the image plane. This is only for simplifying the analysis; all the results can be transformed automatically to the actual case. The system we will be using is depicted in Figure A.1. We define the optical axis in this case to be the perpendicular from the pinhole to the image plane. We introduce a Cartesian coordinate system with the origin at the nodal point and the z-axis aligned with the optical axis and pointing toward the image (Figure A.2). We would like to compute where the image A' of the point A on some object in front of the camera will appear. We assume that nothing lies on the ray from point A to the nodal point O. Let $V=(X,Y,Z)$, the vector connecting O to A, and $V'=(x,y,f)$, the vector connecting O to A', with f the focal length, i.e., the distance of the image plane from the nodal point O, and (x,y) the coordinates of the point A' on the image plane in the naturally induced coordinate system with origin the point of the intersection of the image plane with the optical axis, and the axes x and y parallel to the axes of the camera coordinate system OX and OY. It is trivial to see that

$$x=\frac{fX}{Z},\ y=\frac{fY}{Z} \tag{A.1}$$

Equations (A.1) relate the image coordinates to the world coordinates of a point. Very often, to further simplify the equations we assume $f=1$ without loss of generality.

A.1.2. Orthographic projection

The orthographic projection model seems unrealistic to the eye of the beginner and so we will motivate its use. If, in the perspective projection model, we have a plane that lies parallel to the image plane at $Z=Z_0$, then we define as magnification, mg, the ratio of the distance between two points measured in the image to the distance between the corresponding points on the plane. So, if we have a small interval on the plane $(dX,\ dY,\ O)$ and the corresponding small interval $(dx,\ dy,\ O)$ in the image, then

$$mg=\frac{(dx)^2+(dy)^2}{(dX)^2+(dY)^2}=\frac{f}{Z_0}<1$$

So a small object at an average distance Z_0 will produce an image that is magnified by mg. It is obvious that the

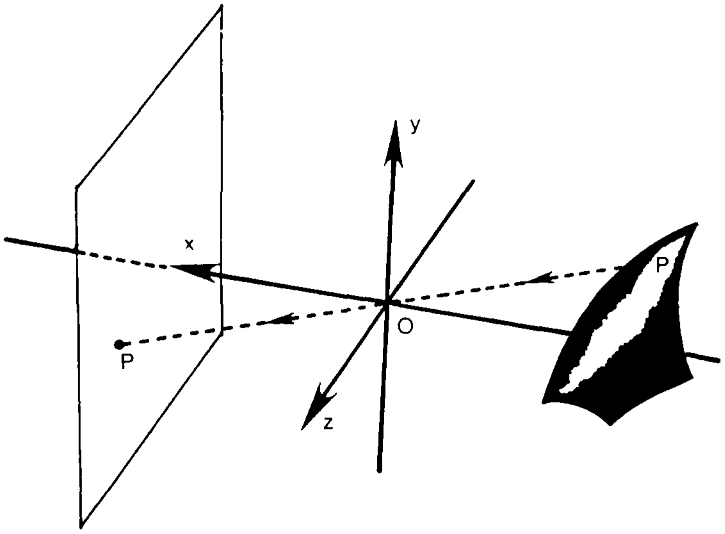


Figure A.1: Perspective projection.

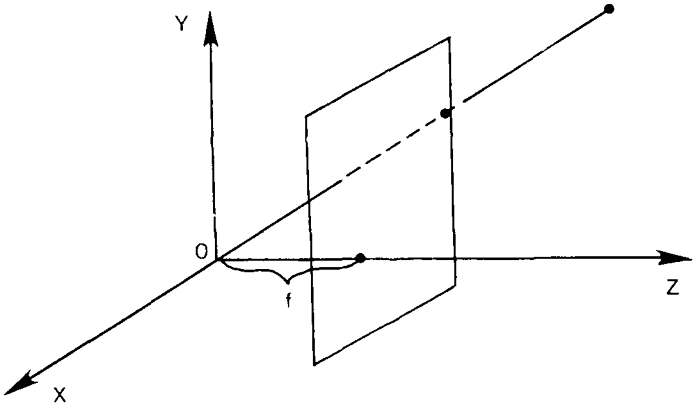


Figure A.2: Perspective projection simplified.

magnification is approximately constant when the depth range of the scene is small relative to the average distance of the surfaces from the camera. In this case we can simply write for the projection (perspective) equations

$$x = mX \quad \text{and} \quad y = mY \tag{A.2}$$

with $m = f/Z_0$ and Z_0 the average value of the depth Z. For our convenience, we can set $m = 1$. Then equations (A.2) are further simplified to the form:

$$x = X \quad \text{and} \quad y = Y \tag{A.3}$$

These equations (A.3) model the orthographic projection model, where the rays are parallel to the optical axis (see Figure A.3). So, the difference between orthography and perspective is small when the distance to the scene is much larger than the variation in distance among objects in the scene. A rough rule of thumb is that perspective effects are significant when a wide angle lens is used, while images taken by telephoto lenses tend to approximate orthographic projection, but of course, this is not exact [Horn, 1986].

A.1.3. Paraperspective projection

The orthographic projection is a very rough approximation of the projection of light on the fovea, but it seems unrealistic for machine vision applications at this point. The perspective projection, a true model, sometimes produces very complicated equations for most of the problems and makes the subsequent analysis very hard. The paraperspective projection is a very good approximation of the perspective, and stands between orthography and perspective. A very similar form of the paraperspective projection was first introduced by Ohta et al. [Ohta et al., 1983]. Let a coordinate system $OXYZ$ be fixed with respect to the camera, with the $-Z$ axis pointing along the optical axis and O the nodal point of the eye. Again we consider the image plane perpendicular to the X axis at the point (0,0,-1) (i.e. focal length $f = 1$, without loss of generality). Consider a small planar surface patch SP on a surface S, with the planar patch obeying the equations $-Z = pX + qY = C$ (see Figure A.4). Under perspective, any point $(X,Y,Z) \in SP$ is projected onto the point

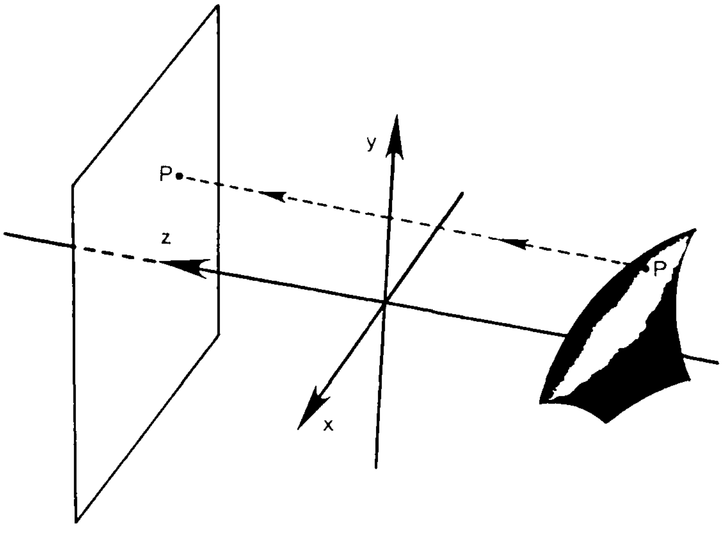


Figure A.3: Orthographic projection.

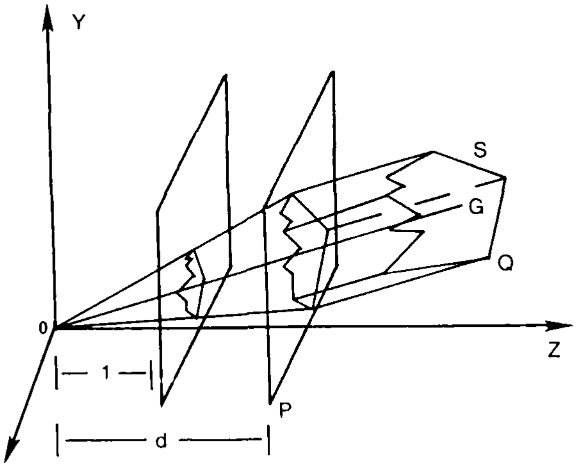


Figure A.4: Paraperspective projection.

$(X/Z,\ Y/Z)$ on the image plane. Let us now see how the small patch SP is projected under the paraperspective projection model.

Consider the plane $-Z=d$, where $-d$ is the Z-coordinate of the center of mass of the region SP. The paraperspective projection is realized by the following two steps:

a) First, the small region SP is projected onto the plane $-Z=d$, which plane is parallel to the image plane and includes the center of mass of the region SP. The projection is performed by using the rays that are parallel to the central projecting ray OG, where G is the center of mass of the region SP.

b) The image on the plane $-Z=d$ is now projected perspectively onto the image plane. Since the plane $-Z=d$ is parallel to the image plane, the transformation is a reduction by a scaling factor $1/d$ (see Figure A.5 which illustrates a cross sectional view of the projection process sliced by a plane which includes the central projecting ray and is perpendicular to the XZ plane). Finally it is clear that the introduced model decomposes the image distortions in two parts: Step (a) captures the foreshortening distortion and part of the position effect, and step (b) captures both the distance and the position effects.

The paraperspective projection process turns out to have nice mathematical properties, since it is an affine transformation.

After having discussed the geometric correspondence between points in the image and points in the scene, we now need to determine the brightness at each image point. But to do that we need some technical prerequisites, which will be found in the next section on intrinsic images.

A.2. Intrinsic Images

In the previous sections we stressed the fact that a very large percentage of modern computer vision is exploiting the recovery of three-dimensional properties (i.e. intrinsic images) from two-dimensional image properties. This section will define mathematically what we mean by intrinsic images, i.e. shape, motion, depth, etc.

Consider again a coordinate system $OXYZ$, fixed with respect to a camera whose nodal point is the origin O and image plane perpendicular to the Z-axis (which is also the optical axis), with focal length f. Consider also the naturally induced image plane xy coordinate system, with origin at the point where the optical axis intersects the image plane and x,y axes parallel to OX and OY respectively. Image coordinates will be denoted by small letters and world coordinates by capital letters. Suppose that the system is imaging a surface S with equation $Z=Z(X,Y)$.

A.2.1. What we mean by shape

We will examine shape under both orthography and perspective projection. Surface orientation is usually represented as the surface normal vector. In intrinsic images, shape

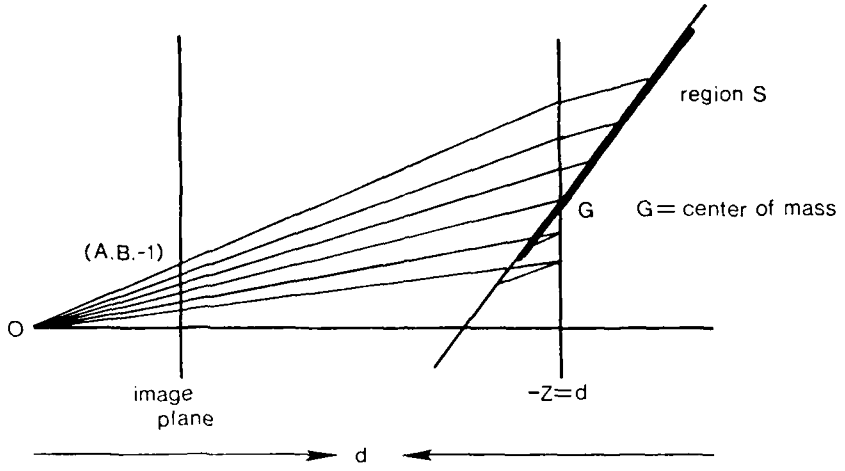


Figure A.5: Cross sectional view of paraperspective.

means the local surface orientation, not some global property of the surface. If the surface is expressed as $Z(X,Y)$ it can be reconstructed from the local shape orientation.

The meaning of shape under perspective

Consider a point $(X,Y,Z) \in S$ whose image under perspective projection is the point $(x = fX/Z,\ y = fY/Z)$. If we say that we know the shape of the object in view at the point (x,y), we mean that we know the surface normal vector n of surface S at the point (X,Y,Z), in particular

$$\bar{n} = \left(\frac{\partial Z}{\partial X}, \frac{\partial Z}{\partial Y}, -1\right) \Big/ \left[\left(\frac{\partial Z}{\partial X}\right)^2 + \left(\frac{\partial Z}{\partial Y}\right)^2 + 1\right]^{1/2}.$$

Suppose now that for every point (x,y) in the image we know the surface normal of the surface patch whose image is the point (x,y). Then, this new image (a surface normal for each point (x,y) of the image) is called the intrinsic shape image. But from only one image we can never hope to compute the exact (X,Y,Z) point, and from it $(\partial Z/\partial X,\ \partial Z/\partial Y,\ -1)$. What we can compute, though, is the quantity $(\partial Z/\partial x,\ \partial Z/\partial y)$, i.e., the gradient of the surface expressed in retinal coordinates. But then, what is the relationship between the gradient in retinal and world coordinates, or in other words, what do we know when we know the quantities $(\partial Z/\partial x,\ \partial Z/\partial y)$?

Consider a point (x,y) on the image and a small displacement in the image $(dx,\ dy)$ from the point, which corresponds to a displacement $(dX,\ dY,\ dZ)$ in the world, on the surface $Z = Z(X,Y)$. Then, from the perspective projection equations, we have

$$dZ = \frac{dx \cdot Z + x\ dZ}{f} \quad \text{and} \quad dY = \frac{dy \cdot Z + y\ dZ}{f}$$

Now, given that $Z(X + dX, Y + dY) = Z(x + dx, y + dy)$, and expanding both sides of this equation in a Taylor series and ignoring the higher order terms, we get

$$\frac{\partial Z}{\partial X}\frac{Z}{f - x\frac{\partial Z}{\partial X} - y\frac{\partial Z}{\partial Y}}dx + \frac{\partial Z}{\partial Y}\frac{Z}{f - x\frac{\partial Z}{\partial X} - y\frac{\partial Z}{\partial X}}dy$$

$$= dx\frac{\partial Z}{\partial x} + dy\frac{\partial Z}{\partial y}$$

from which

$$\frac{\partial Z}{\partial x} = \frac{Z\frac{\partial Z}{\partial X}}{f - x\frac{\partial Z}{\partial X} - y\frac{\partial Z}{\partial Y}} \quad \text{and} \quad \frac{\partial Z}{\partial y} = \frac{Z\frac{\partial Z}{\partial Y}}{f - x\frac{\partial Z}{\partial X} - y\frac{\partial Z}{\partial Y}}$$

From the above equations it is easy to see that if $\delta Z/\delta X$, $\delta Z/\delta Y$ are known, then the quantity

$$\frac{Z(x + dx, y + dy)}{Z(x,y)}$$

is computable. But this means that if the surface normals are known indexed by retinal coordinates, then the depth function $(Z(x,y))$ can be computed up to a constant factor. In other words, if shape is known, then for any two points (x_i,y_i) and (x_j,y_j) on the image, we know the ratio

$$\frac{Z(x_i,y_i)}{Z(x_j,y_j)}.$$

So, an object whose shape we know under perspective projection can be small and near the camera or large and far away.

The meaning of shape under orthography

Under orthographic projection, the image coordinates of a point are equal to the corresponding 3-D coordinates, i.e. $(x,y) = (X,Y)$. So

$$\left(\frac{\partial Z}{\partial X}, \frac{\partial Z}{\partial Y}\right) = \left(\frac{\partial Z}{\partial x}, \frac{\partial Z}{\partial y}\right).$$

Obviously, if we know shape in this case, since

$$Z(x + dx, y + dy) - Z(x,y) = \frac{\partial Z}{\partial x}dx + \frac{\partial Z}{\partial y}dy + (\text{h.o.t.}),$$

we know that the depth function can be computed up to constant additive term. So, if we know shape under orthography, we know exactly the object in view, but we do not know its depth.

Other representations for shape

We have stated that the surface normal

$$\frac{(p, q, 1)}{(p^2 + q^2 + 1)^{\frac{1}{2}}}$$

with $p = \delta Z / \delta X$, $q = \delta Z / \delta Y$ at a point of a surface $Z = Z(X,Y)$ represents the shape. This

$$(p,q) = \left(\frac{\partial Z}{\partial X}, \frac{\partial Z}{\partial Y} \right)$$

is not the only representation. Obviously shape is nothing but a direction in three-dimensional space, and so there are many representations for it. The ones that we will use quite often in this paper are, with the exception of the gradient that we have already analyzed, the following:

a) Coordinates (a,b,c) on the Gaussian sphere.

b) Latitude and longitude angles, say, (θ,ρ).

c) Slant and tilt. Slant is the tangent of the latitude angle and tilt is the longitude angle. The notation for (slant, tilt) is (σ,τ). The slant and tilt are polar versions of the (p,q) coordinates.

The relationship among these different representations is given by the following equations:

$$\sigma = \tan\theta = \sqrt{(p^2 + q^2)}$$

$$p / q = \tan\phi = \tan t$$

Finally, if (a,b,c) are the coordinates on the Gaussian sphere, then

$$(a,b,c) - \left(\frac{p}{k}, \frac{q}{k}, \frac{1}{k} \right) \text{ with } k - (p^2 + q^2 + 1)^{\frac{1}{2}}$$

A.2.2. What we mean by retinal motion

If the object in view is moving with a general motion, or if the camera is moving, or if both move, then the image is moving too. Let the retinal velocity at an image point be (u,v). The resulting vector field (the velocity of every image point) is called the retinal motion field or optic flow field. This flow field is an intrinsic retinal motion image.

A.2.3. What we mean by depth

Consider again a surface S with equation $Z = Z(X,Y)$ in front of the camera. Every point (x,y) in the image is the projection of a point $(X,Y,Z) \in S$. If for every point (x,y) on the image we know the Z coordinate (depth) of the corresponding 3-D point (X,Y,Z), then we know exactly where the surface is with respect to the camera coordinate system. The resulting image (for every point in the image there corresponds a number (depth) of the corresponding 3-D point) is called the intrinsic depth image.

A.2.4. Intrinsic parameters that are not images

There exist intrinsic parameters which do not correspond to every point in the image. These are global constants and every point in the image is in some relation to them. Examples of these parameters are the 3-D motion and lighting direction parameters.

3-D motion parameters

If an object moves in front of a camera with a general motion, then this motion can be considered as the sum of a translation (U,V,W) and a rotation (A,B,C). These six parameters will be called motion parameters.

Lighting direction parameters

Consider again a surface in front of a camera, illuminated by a light source in the direction (l_x,l_y,l_z), with respect to the camera coordinate system. The direction (l_x,l_y,l_z) is called the lighting or illuminant direction.

A.2.5. A synopsis

Up to this point we have defined mathematically so-called intrinsic parameters. These are shape, depth, retinal motion, 3-D motion, and light source direction. This of course does not mean that these are the only intrinsic parameters. There can be many more but the ones that we described here are the ones which we (and contemporary research) think are *the most important for the perception of* the outside world. Again, we do not want to get involved in philosophical arguments about why these intrinsic parameters are important to compute for visual perception. The shape of objects is important for the recognition of objects that we see, the depth of objects is important for our interaction with the environment (picking up things), retinal motion is important for understanding discontinuities and segmenting the environment as well as for the computation of the 3-D motion which is important for navigation and for understanding the motion of objects in our environment as well as for avoiding moving objects.

There may very well be other important intrinsic parameters that we haven't discovered yet. There may also be no more intrinsic parameters of interest. Further research will uncover the truth of this matter.

A.3. Brightness at Every Image Point

In this section we analyze how the brightness at every image point is determined. The amount of light reflected by a surface element depends on its microstructure, on its optical properties and on the distribution and state of polarization of the incident illumination. For several surfaces, the fraction of incident illumination reflected in a particular direction depends only on the surface orientation. The characteristics of the reflectance of such a surface can be represented as a function $f(i,g,e)$ of the angles i = incident, g = phase and e = emergent, as they are defined in Figure A.6.

The reflectance function $f(i,g,e)$ determines the ratio of surface radiance to irradiance measured per unit surface area, per unit solid angle, in the direction of the viewer. If we want to be precise, we should specify the quantities and units used to define the required ratio. Here it is sufficient to point out the role that surface orientation plays in the determination of the angles i and g.

Consider the example of perfect specular (mirror-like) reflection. In this case, the incident angle equals the emergent angle and the incident, emergent and normal vectors lie on the same plane $(g \quad i + e)$. So, the reflectance function is

$$f(i,e,g) = \begin{vmatrix} 1 \text{ if } i \quad e \text{ and } i + e \quad g \\ 0, \text{ otherwise} \end{matrix}$$

The interaction of light with surfaces of varying roughness and composition of material leads to a more complicated distribution of reflected light. Surface reflectance characteristics can be determined empirically, derived from models of surface microstructure or derived from phenomenological models of surface reflectance. The *most widely used model* of surface reflectance is given by the function $f(i,e,g) \quad \rho\cos i$, where ρ is a constant depending on the specific surface. This reflectance function corresponds to a phenomenological model of a perfectly diffuse (Lambertian) surface which appears

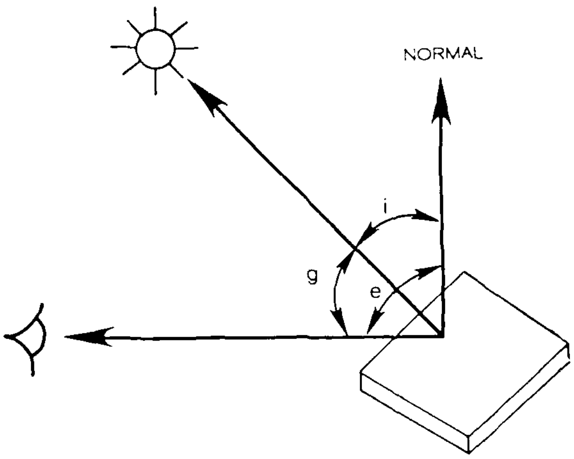


Figure A.6: Reflectance model.

equally bright from all viewing directions; the cosine of the incident angle accounts for the foreshortening of the surface as seen from the source.

The *surface normal vector* relates surface geometry to image irradiance because it determines the angles i and e appearing in the surface reflectance function $f(i,e,g)$. In orthographic projection, the viewing direction and so the phase angle g is constant for all surface elements. So, for a fixed light source and viewer geometry and fixed material, the ratio of scene radiance to scene irradiance depends only on *the surface normal vector. Furthermore, suppose that each* surface element receives the same irradiance. Then, the scene radiance and hence image intensity depends only on the surface normal vector. A reflectance map $R(p,q)$ determines image intensity as a function of p and q (where $(p, q, -1)/\sqrt{(p^2+q^2+1)}$ is the surface normal vector). Using a reflectance map, an image irradiance equation can be written as $I(x,y) = R(p,q)$, where $I(x,y)$ is the intensity at the image point (x,y) and $R(p,q)$ is the corresponding reflectance map.

A reflectance map provides a uniform representation for specifying the surface reflectance of a surface material for a particular light source, object surface and viewer geometry. A comprehensive survey of reflectance maps derived for a variety of surface and light source conditions has been given by Horn [Horn, 1977]. Furthermore, a unified approach to the specification of surface reflectance maps has been given in [Horn and Sjoberg, 1981].

Expressions for $\cos i$, $\cos e$ and $\cos g$ can be easily derived from the surface normal vector $(p, q, -1)$ and the light source vector $(p_s, q_s, -1)$ and the vector $(0, 0, -1)$ which points in the direction of the viewer. For a Lambertian reflectance function we get

$$R(p,q) = \frac{\rho(1 + p\,p_s + q\,q_s)}{\sqrt{(1+p^2+q^2)}\sqrt{(1+p_s^2+q_s^2)}}$$

So, for a Lambertian surface, the intensity $I(x,y)$ at a point (x,y) of the image is given by

$$I(x,y) = \frac{\rho(1 + p\,p_s + q\,q_s)}{\sqrt{(1+p^2+q^2)}\sqrt{(1+p_s^2+q_s^2)}}$$

with ρ the albedo constant and $(p, q, -1)$ and $(p_s, q_s, -1)$ the surface normal at the point whose image is the point (x,y) and the light source direction respectively, under orthographic projection. Under perspective projection, the model is not yet known exactly.

Appendix B

Proposition:

Let a coordinate system $OXYZ$ be fixed with respect to the left camera, with the Z axis pointing along the optical axis. Let the nodal point of the right camera be the point (Sx, Sy, O) and its image plane be parallel to the previous one, at $Z = 1$. Consider a region Ω in the world plane $Z = pX + qY + c$ having area S_w. Let S_1, S_2 be the areas of the perspective projection of Ω in the left and right cameras respectively and let A_1, A_2 be their respective centroids. Then we have

$$\frac{S_2}{S_1} = \frac{1 - pA_2 - qB_2}{1 - pA_1 - qB_1}$$

Proof:

If (x',y') denote the coordinates on the surface plane and (x_i,y_i) $i = 1, 2$ the coordinates on image planes 1 and 2 respectively and assuming that the foci of the two cameras are at the points (0,0) and (S_x,S_y) respectively with coinciding image planes $z = -1$, we have

$$(A_i,B_i) = \frac{\int_{\Omega_i} (x_i,y_i)\,dx_i\,dy_i}{\int_{\Omega_i} dx_i\,dy_i} \quad i = 1, 2$$

where Ω_i $i = 1, 2$ are the two contours. Consequently

$$(A_i,B_i) = \int_{\Omega_i} (x_i,y_i) \left| \frac{\partial(x_i,y_i)}{\partial(x',y')} \right| dx'\,dy' \quad i = 1, 2$$

where $J_i = \dfrac{\partial(x_i, y_i)}{\partial(x', y')}$ $i = 1, 2$ the Jacobian of the transformation from the world to each one of the image planes. We will consider only cases in which the orientation of the plane may be continuously transformed to $p = q = 0$ without passing through an orientation where it appears edge-on in one of the cameras, i.e. both cameras are looking at the front of the plane, not the back. In this case, the J_i's are positive and we may drop the absolute value signs. So, $J_i = \dfrac{(1 - px_i - qy_i)^3}{c_i^2(1 + p^2 + q^2)^{1/2}}$ $i = 1, 2$ with $c_1 = c$, $c_2 = c + pS_x + qS_y$. Note that since $-z = \dfrac{c_1}{1 - px_1 - qy_1} = \dfrac{c_2}{1 - px_2 - qy_2}$, we have

$$J_i = \frac{c_i}{(-z)^3(1 + p^2 + q^2)^{1/2}}$$

So,

$$\frac{1 - pA_2 - qB_2}{1 - pA_1 - qB_1} = \frac{1 - p\left(\dfrac{\int_\Omega x_2 J_2\, dx'\, dy'}{\int_\Omega J_2\, dx'\, dy'}\right) - q\left(\dfrac{\int_\Omega y_2 J_2\, dx'\, dy'}{\int_\Omega J_2\, dx'\, dy'}\right)}{1 - p\left(\dfrac{\int_\Omega x_1 J_1\, dx'\, dy'}{\int_\Omega J_1\, dx'\, dy'}\right) - q\left(\dfrac{\int_\Omega y_1 J_1\, dx'\, dy'}{\int_\Omega J_1\, dx'\, dy'}\right)}$$

or

$$\frac{1 - pA_2 - qB_2}{1 - pA_1 - qB_1} = \frac{\int_\Omega J_1\, dx'\, dy' \int_\Omega (1 - px_2 - qy_2) J_2\, dx'\, dy'}{\int_\Omega J_2\, dx'\, dy' \int_\Omega (1 - px_1 - qy_1) J_1 dx'\, dy'}$$

or

$$\frac{1 - pA_2 - qB_2}{1 - pA_1 - qB_1} = \frac{\dfrac{c_1}{(1 + p^2 + q^2)^{1/2}} \int \left(\dfrac{1}{-z}\right)^3 dx'\, dy' \int \dfrac{c_2}{(-z)} \dfrac{c_2}{(1 + p^2 + q^2)^{1/2}} \dfrac{1}{(-z)^3} dx'\, dy'}{\dfrac{c_2}{(1 + p^2 + q^2)^{1/2}} \int \dfrac{1}{(-z)^3} dx'\, dy' \int \dfrac{c_1}{(-z)} \dfrac{c_1}{(1 + p^2 + q^2)^{1/2}} \dfrac{1}{(-z)^3} dx'\, dy'}$$

or

$$\frac{1 - pA_2 - qB_2}{1 - pA_1 - qB_1} = \frac{c_2}{c_1} = \frac{c + pS_2 + qS_y}{c} \qquad (1)$$

On the other hand,

$$S_1 = \int_\Omega \left| \frac{\partial(x_i, y_i)}{\partial(x', y')} \right| dx'\, dy'$$

$$S_2 = \int_\Omega \left| \frac{\partial(x_2, y_2)}{\partial(x', y')} \right| dx'\, dy'$$

Also,

$$J_1 = \frac{c}{(-z)^3(1 + p^2 + q^2)^{1/2}} \quad \text{and}$$

$$J_2 = \frac{c + pS_x + qS_y}{(-z)^3(1 + p^2 + q^2)^{1/2}}$$

So, $$\frac{S_2}{S_1} = \frac{c + pS_x + qS_y}{c} \, \frac{\int \left(-\dfrac{1}{z}\right)^3 dx'\, dy'}{\int \left(-\dfrac{1}{z}\right)^3 dx'\, dy'} \quad \text{or}$$

$$\frac{S_2}{S_1} = \frac{c + pS_x + qS_y}{c} \qquad (2)$$

(1), (2) prove the claim.

COMPUTATION OF MOTION IN DEPTH PARAMETERS: A FIRST STEP IN STEREOSCOPIC MOTION INTERPRETATION.

Poornima Balasubramanyam
M.A. Snyder

Computer Vision Research Laboratory
Department Of Computer and Information Science
University Of Massachusetts
Amherst, Ma. 01003

Abstract

This paper is motivated by the need to obtain the three parameters of absolute 3D motion in depth of objects in a dynamic scene using visual information. *Motion in depth (MID)* parameters refer to the following three components of the object or camera motion, i.e., the single component of translation in depth (parallel to the line of sight) and the two components of rotation in depth (or rotational components that are not about the line of sight). In this paper we show that use of *stereoscopic motion* enables the MID parameters to be computed in a quick and robust manner for a stereo camera system moving through an environment that may have several independently moving rigid bodies. It has been shown elsewhere that use of temporal binocular imagery permits the formulation of the ratio of the relative optic flow between a stereo pair of images and the disparity between them as a *linear function* of the image coordinates. The *coefficients of this linear function are the translation in* depth and rotation in depth components which are precisely the MID parameters being computed in our approach.

1. Introduction

Motivation

The human visual system exhibits remarkable skill in detecting the *translation in depth* of moving objects. Consider, for instance, the following two commonly performed tasks - catching a ball that is thrown toward the observer with a rotational spin component contributing to the motion, and navigating around obstacles moving toward or away from the observer. The observer has to make rapid judgements about the translation in depth of the object in question. Determination of the complete trajectory of the object involves determination of all the parameters of motion and is less *important* to the immediate needs of the observer. It is possible to make such judgements of translation in depth even when the object has other motion attributes, such as rotations about different axes contributing to the actual trajectory. Perception of the translation in depth of the spinning ball moving toward the observer is one such example. Hence, we believe that the translation in depth of objects is a particularly useful motion parameter to compute for both navigation and moving obstacle avoidance.

Also of significant interest to us is the psychophysical evidence that demonstrates the perception of *rotation in depth* by human observers. There are specific empirical findings [9,17,19] that are concerned with the ability of human observers to perceive three dimensional relationships on the basis of rotation in depth. The emphasis in the above work has been on both the perception of the direction of rotation as well as on the overall impression of depth or coherence elicited by the rotating objects. Computational models dealing with the structural information that can be extracted from such an image transformation have been developed [28,29]. We believe that the perception of rotation in depth of objects in the image can be used to guide further semantic analysis of the scene in order to yield structural as well as motion descriptions of semantically meaningful objects in the scene.

Motivated by this, we propose here a model of *motion in depth* computation that permits the presence of general motion of both the camera and multiple objects in the environment.

Approach

Motion in depth (MID) parameters refer to the following three components of the object or camera motion: the single component of translation in depth (along the line of sight) and the two components of rotation in depth (rotations that are not about the line of sight).

The model uses stereo time-varying optic flow fields to perform a fast and robust computation of the parameters that specify the absolute MID values. It is based on the concept of stereoscopic motion, described by Beverly and Regan [8]. *Stereoscopic motion* refers to the relative motion present between the stereo pair of a temporal image sequence. Psychophysical evidence [8] indicates that stereoscopic motion is used by human observers to extract the translation in depth of objects in the scene.

In the proposed model, therefore, we consider the *dynamics* of a scene as viewed by a stereo pair of cameras. This is distinct from the analysis of the static imagery of the stereo pair at one time instant to obtain the static disparity between the two images. The model assumes for its input, the existence of the static disparity field corresponding to the first stereo pair of the two frame stereo sequence along with the optic flow computed seperately for each of the two stereo image sequences. These are used to determine the relative image change due to motion in the temporal binocular image sequence. Issues dealing with the correspondence problem in stereo and motion analysis are bypassed for the purposes of the current work. Hence, the model proposes an integrated *interpretation* of the disparity and optic flow information as opposed to an integrated analysis of stereo and motion that deals with correspondence issues. The latter approach has been adopted in other work in this area [13,30]. These and other related work have been discussed in the following section.

It was shown by Waxman and Duncan [30] that use of temporal binocular imagery permits the formulation of a linear relationship between the relative flow and disparity values and the MID parameters. This was used to establish correspondence between stereo pairs of images. We believe this to be insufficient use of the information that is available from the linear relationship and hence propose to use this formulation as the theoretical basis for our approach to computing the MID parameters of object or sensor motion for a sensor travelling through an environment that may have independently moving objects.

A two-stage global technique is employed here to compute the three motion in depth parameters. In the first stage, we employ the segmentation technique developed by Adiv [1,2] as the initial stage of his algorithm for interpreting monocular optic flow fields. We perform this segmentation in order to obtain a 2-D grouping of the optic flow field. In the second stage, segments from the first stage are grouped into regions that correspond to the same set of three MID parameters by employing a least squares minimization technique on the relative optic flow between the stereo pair of image sequence.

Note that such a two-step view of 3-D motion interpretation essentially follows the viewpoint of Adiv [1]. Performing segmentation on the 2D image plane provides a method of restricting the actual 3D interpretation mechanism, (i.e., the global minimization step in the current model), to a semantically relevant set of flow vectors and thus contributes to the robustness of the entire algorithm. A direct attempt at interpreting the 3D motion from the 2D flow without an intermediate grouping would necessarily be a local analysis of the flow and hence be very susceptible to noise. On the other hand, it is important to note that this segmentation step uses an approximation to determine the grouping. Hence, it is important not to use this step to directly compute the values of the 3D motion parameters, but use this grouping as a mask on the *relative flow field* in order to determine the values of the MID parameters without making any surface approximations.

MID parameters can be obtained from a general 3D motion computation algorithm using monocular imagery, but these algorithms are computationally intensive and do not permit quick reliable computation. One of the chief reasons for this is that the flow information to the depth and motion parameters via nonlinear equations. The main advantage provided by the proposed integrated interpretation of stereo and motion information for extracting motion in depth is that stereo information provides additional constraints that make it possible to formulate a *linear* relationship between the data in the flow and disparity fields and the motion in depth parameters. This, in turn, makes it possible to devise a direct computation without hypothesizing any of the motion parameters such as in [1,20]. This can conceivably be used to directly compute the remaining three motion parameters, again, without employing a hypothesize and test scheme. Hence, extracting all the motion parameters would become a problem of handling two sets of linear functionals, and we plan to demonstrate this in future.

In Section 2, we review existing techniques for dealing with the problem of integrating stereo and motion information. The assumptions and limitations of these techniques are discussed. We also briefly review current motion interpretation research. In Section 3, we formulate the theoretical model. In Section 4, we describe the algorithm and discuss the decisions that were made in the development and implementation of the algorithm. Experiments based on simulated data are described in Section 6. These experiments demonstrate the generality of the algorithm. In future work, we plan to demonstrate the algorithm on real data. Anandan's algorithm [4,5] which has shown state of the art performance on real imagery, will be used to extract the optic flow fields. A modified version of the same algorithm can be used to extract the disparity values as well. In Section 7, we summarize the approach and major results, and discuss directions for future research.

2. Literature Survey

In this section we examine the approaches taken by several authors to the problem of interpreting temporal binocular imagery. The research over the years has chiefly been on the seperate interpretation of optic flow in monocular imagery or on static stereo analysis. An extensive review of work on the interpretation of monocular optic flow is given in [7], and more recent work includes that of [3,31].

Any method for the interpretation of monocular optic flow fields is limited by the fact that the flow value at a point in the image is a nonlinear function of the six parameters of motion and the depth of the corresponding environmental point. Dealing with this imposes restrictions on any interpretation technique. For instance, iterative methods such as [10,20,26] need good initial guesses. Sensitivity to noise is reported by much of the earlier research e.g.,[10,20,25,27]. Some techniques, such as [15] and [11], either assume restricted motions such as pure translation or assume that one component of the motion such as the rotation may be known and can be subtracted out prior to the computation. Others deal with a stationary environment and moving camera, disallowing the possibilty of multiple independently moving objects [24].

Some of the more successful general methods that deal with multiple independently moving objects, as well as the presence of general motion, can be found in [1,2] and [31]. Good results in determining 3 D motion and object structure for simulated data have been reported in [31], although computation of the local derivatives of the flow may be highly sensitive to noise.

The work of **Adiv** [1,2] appears to be robust. However, the technique adopted for the computation of the motion parameters is computationally intensive and does not permit quick evaluation of at least some of the motion parameters that would be important in a practical context. This is a defect that is inherent to any method that deals with monocular imagery due to the underconstrained nature of the relationship between the flow field and the information desired from it. In the case of the method adopted in [1,2], for instance, the motion parameters are computed only at the end of a search of a sampling of possible translation directions and corresponding optimal rotation parameters, an approach that is essentially a bottom-up hypothesize and test scheme in practice.

Any method that is directed toward quickly extracting motion parameters, while retaining the ability to deal with multiple objects and general motion has to deal with the fact that monocular imagery with just two frames provides underconstrained information. There is a need to look for additional sources of information that will provide more constraints for computing the motion parameters. This leads to the use of

- multiple frames of monocular imagery, and
- stereo time-varying imagery

We now briefly review the previous work that addresses the latter aspect of integrating stereo and motion analysis. Techniques have been formulated to facilitate the solution to the correspondence problem for both stereo and motion as well as to address interpretation issues.

The first use of the term *stereoscopic motion* was in the work of **Regan, et al.** [21]. They addressed the problem of using visual information to recover motion in depth of objects, and provide evidence to support models of neural organizations in the human visual system for detecting the motion in depth of objects. They propose the presence of neural "filters" that are sensitive to the relative velocities in the left and right retinal images and are thereby selectively sensitive to the *direction* of motion in depth of objects in the visual field. These motion-in-depth detectors are thus viewed as binocularly driven channels that process the changing disparity. Our computational model has been motivated by the psychophysical evidence that strongly supports such a formulation. Also of interest to us is pychophysical evidence in [21] for the following -

- changing disparity grows relatively more effective as the velocity increases, and
- changing disparity grows relatively more effective as the inspection time increases.

In future work, we will examine the relevance of these conclusions to the proposed model.

Other work of Regan [22] indicates that the human visual system possesses sensitivity to four kinds of relative motion, namely,

- the velocity difference between two points in one retinal image along the line joining the two points,
- the velocity difference between the two points in one retinal image perpendicular to the line joining them,
- rotary motion, and
- the ratios of the velocities of the left and right retinal images of an object moving in depth.

We note that the latter sensitivity may be used to recover object motion in depth. An interesting alternate viewpoint provided here is that these filters may provide physiological means of analysing the local flow in the retinal images in order to recover specific information about the environment.

The approach taken by **Richards** [23] to integrate stereo and motion information uses both to extract three dimensional information about the environment in a manner that results in a unique solution for the *object structure*. The problem with solving for structure from pure motion information is that any algorithm needs to deal with second order equations relating the flow to the structure. This will result in the possibility of multiple interpretations of object configurations. It is required that these be removed by disambiguating the possible solutions. The problem noted with stereopsis is that the correct configuration of objects can be determined only if the fixation distances are known. This is because the same configuration at different distances will result in different angular disparities. Motion information can correctly interpret the angular relations between objects, thus making knowledge of the fixation distance unnecessary. The approach uses stereopsis to provide information which disambiguates the multiple interpretations found with pure motion since stereo can provide absolute depth information. We note that this work deals purely with the problem of extracting the structure of the environment and does not deal with the use of stereo and motion information to extract the motion parameters of objects in the environment or of the camera.

The approach of **Jenkin** [13] uses stereo and motion information in a prediction-correction mechanism to facilitate the solution to the ***correspondence problem*** both in stereo and motion analysis. The algorithm predicts the position that a point in the current frame might correspond to, using the previous motion of that point, i.e., by using the previous frames. Hence, the stereopsis correspondence problem is simplified since the search area is restricted to that predicted by the analyses of the previous frame pairs. We note that the algorithm addresses only the correspondence problem for both stereo and motion, and not the interpretation of stereo or motion information.

A unified approach to the analysis of stereo and motion data is given by **Waxman and Duncan** [30]. They show that there is a correlation between the stereo disparity and the relative flow between the stereo pair of an image sequence. This result is used to establish correspondence in the context of local support from neighbourhoods. It is proposed that in the analysis of time-varying stereo imagery, after the initial correspondence is established, matching for subsequent images need be performed only at the peripheral regions of the image as well as around occluding boundaries. We note that this work approaches the integration of stereo and motion information from the viewpoint of facilitating the correspondence problem.

An entirely different approach using two frames of stereo imagery is taken by **Huang and Blostein**, [12]. They estimate the rigid body rotation and translation parameters by matching 3-D points determined at two time instances from stereo information. This 3-D matching problem can be solved by considering geometric relationships that should be preserved under rigid body displacements. The 2-D matching problem continues to persist for the stereo matching, while the motion estimation algorithm needs to deal with appropriate mechanisms for 3-D matching.

Mutch [18] describes a technique to recover the translation in depth of a moving object point, using the concept of stereoscopic motion as described by [8]. The change in the location of the image of an object point is defined by its "change vector." The relative difference between two change vectors in the left and right image sequences corresponding to a translating object point is such that their relative orientation and magnitude leads to the perception of certain 3-D properties of the object, including its translation in depth. The direction of the translation in depth as well as the position of the impact point can be determined from this method. We note the approach requires that in the case of general motion by the object point, the rotational component first be removed from the change vector. This work is of interest since it uses stereoscopic motion in a computational context to determine the translation in depth of objects.

We give a more general approach to the problem of interpreting stereo and motion information by being able to deal with general motion and the presence of multiple independently moving objects. Also, we do not restrict the point of interest to a single object point, such as the centroid in [18], but develop the technique for dense flow and disparity fields. This is more robust since more global information is allowed in the computation. Finally, we extract translation as well as rotations in depth of the camera and the objects

3. Theoretical Formulation Of The Model

In this section we develop the mathematical framework for our approach. We first consider monocular flow analysis, and then generalize binocular flow.

Monocular Flow Analysis

Given the optic flow on the image plane, we can relate the values of the components of the flow field at every point in the image to the 3-D motion parameters and depth of the environmental point that projects to this image point.

Let us consider a cartesian coordinate system (X, Y, Z) that is fixed with respect to the camera with the focal length normalized to a distance of 1 from the origin, O, to the image plane. Let (x, y) represent the image coordinate system. The perspective projection of an environmental point, (X, Y, Z), on the image plane is then given by

$$x = X/Z, \qquad (1)$$

$$y = Y/Z \qquad (2)$$

We now consider the motion relative to the camera of a rigid object in the environment. Let P be the position vector of some point on the object, with camera coordinates given by (X, Y, Z) (see Figure 1). Since the object is rigid, the instantaneous velocity, $\dot{P}$, of P can be represented as an combination of an infinitesimal rotation Ω, and an infinitesimal translation T. Thus,

$$\dot{P} = \Omega \times P + T. \qquad (3)$$

In components, this becomes :

$$\begin{pmatrix} \dot{X} \\ \dot{Y} \\ \dot{Z} \end{pmatrix} = \begin{pmatrix} \Omega_Y Z - \Omega_Z Y + T_X \\ \Omega_Z X - \Omega_X Z + T_Y \\ \Omega_X Y - \Omega_Y X + T_Z \end{pmatrix}.$$

The corresponding image point (x, y) has an image velocity given by (α, β), where

$$\begin{pmatrix} \alpha \\ \beta \end{pmatrix} \overset{\mathrm{def}}{=} \begin{pmatrix} \dot{x} \\ \dot{y} \end{pmatrix} = \frac{1}{Z^2} \begin{pmatrix} \dot{X}Z - \dot{Z}X \\ \dot{Y}Z - \dot{Z}Y \end{pmatrix}.$$

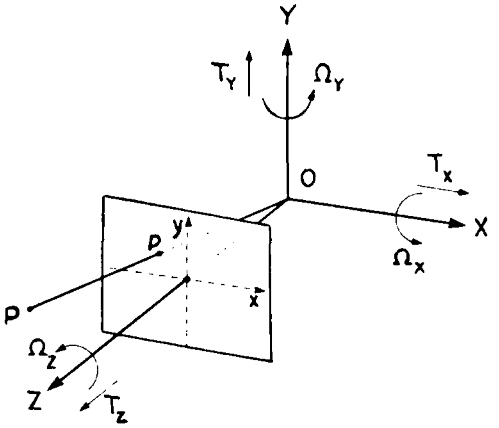


Figure 1: Monocular Camera Configuration

which becomes, from (3) :

$$\alpha = -\Omega_X xy + \Omega_Y(1 + x^2) - \Omega_Z y + (T_X - T_Z x)/Z, \qquad (4)$$

$$\beta = -\Omega_X(1 + y^2) + \Omega_Y xy + \Omega_Z x + (T_Y - T_Z y)/Z. \qquad (5)$$

We therefore see that the flow equations (4,5) are 2^{nd} order functions of (x, y).

Binocular Flow Analysis

We now extend the above derivation to the case of a stereo pair of cameras :

The two image planes are coplanar,and perpendicular to the ground plane. We assume that the focal points lie along the same horizontal line (see Figure 2). The analysis follows that of [30].

Let us denote the cartesian coordinate system for the left camera by (X_l, Y_l, Z_l) and that for the right camera by (X_r, Y_r, Z_r). Let the horizontal displacement between the two focal points, O_l and O_r (see Figure 2) be b. We denote entities with respect to the left and right cameras by the subscripts l and r respectively. The final formulations are with respect to the left coordinate system.

Consider a point at (X_l, Y_l, Z_l). The projection of this point on the left image plane is given by (x_l, y_l). Similarly, the point

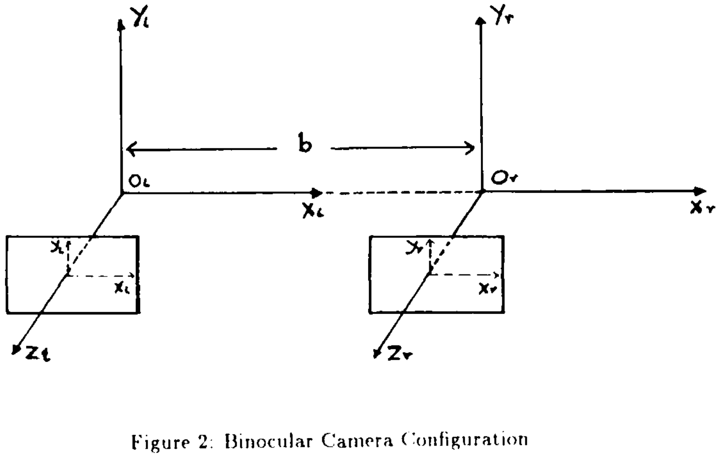


Figure 2: Binocular Camera Configuration

is at a position (X_r, Y_r, Z_r) with respect to the right coordinate system, and it projects on the right image plane at (x_r, y_r). Let the disparity between the corresponding image points be given by δ, where

$$\delta \overset{\text{def}}{=} x_l - x_r = b/Z, \tag{6}$$

since

$$Z_l = Z_r \equiv Z. \tag{7}$$

Also note that

$$y_l = y_r \equiv y. \tag{8}$$

Let us now consider the flow equations for the two cameras. From equations (4,5), we have with respect to the left camera

$$\begin{aligned} \alpha_l(x_l, y_l) &= -\Omega_{X_l} x_l y_l + \Omega_{Y_l}(1 + x_l^2) - \Omega_{Z_l} y_l + \\ &\quad (T_{X_l} - T_{Z_l} x_l)/Z, \qquad (9) \\ \beta_l(x_l, y_l) &= -\Omega_{X_l}(1 + y_l^2) + \Omega_{Y_l} x_l y_l + \Omega_{Z_l} x_l + \\ &\quad (T_{Y_l} - T_{Z_l} y_l)/Z, \qquad (10) \end{aligned}$$

and with respect to the right camera,

$$\begin{aligned} \alpha_r(x_r, y_r) &= -\Omega_{X_r} x_r y_r + \Omega_{Y_r}(1 + x_r^2) - \Omega_{Z_r} y_r + \\ &\quad (T_{X_r} - T_{Z_r} x_r)/Z. \qquad (11) \\ \beta_r(x_r, y_r) &= -\Omega_{X_r}(1 + y_r^2) + \Omega_{Y_r} x_r y_r + \Omega_{Z_r} x_r + \\ &\quad (T_{Y_r} - T_{Z_r} y_r)/Z. \qquad (12) \end{aligned}$$

But from the geometry we know that

$$\Omega_l = \Omega_r \equiv \Omega \tag{13}$$

and

$$T_r = T_l - \Omega_l \times b\hat{x}. \tag{14}$$

where $\hat{x}$ is the unit vector pointing from O_l to O_r. Hence we can rewrite the flow equations (11,12) for the right image plane using equations (13, 14). We then have with respect to the left coordinate system,

$$\begin{aligned} \alpha_r(x_l + \delta, y_l) &= [-\Omega_{X_l}(x_l y_l + y_l) + \Omega_{Y_l}(1 + x_l + x_l^2) - \\ &\quad \Omega_{Z_l} y_l + (T_{X_l} - tzl - T_{Z_l} x_l)/Z]\delta \qquad (15) \\ \beta_r(x_l + \delta, y_l) &= -\Omega_{X_l}(1 + y_l^2) + \Omega_{Y_l} x_l y_l + \Omega_{Z_l} x_l + \\ &\quad (T_{Y_l} - T_{Z_l} y_l)/Z. \qquad (16) \end{aligned}$$

The two components of the *relative flow* between the two images are thus given by

$$\begin{aligned} \Delta\alpha &= \alpha_r(x_l + \delta, y_l) - \alpha_l(x_l, y_l) = \\ &\quad [\Omega_{Y_l} x_l - \Omega_{X_l} y_l - T_{X_l}/Z]\delta, \qquad (17) \\ \Delta\beta &= \beta_r(x_l + \delta, y_l) - \beta_l(x_l, y_l) = 0. \qquad (18) \end{aligned}$$

We can now write the ratio of the relative flow between the two image points to the disparity, expressed with respect to the left camera coordinate system, as :

$$\begin{aligned} \frac{\Delta\alpha}{\delta} &= \Omega_Y x_l - \Omega_X y_l - T_Z/Z \qquad (19) \\ \frac{\Delta\beta}{\delta} &= 0. \qquad (20) \end{aligned}$$

We refer to the vector $(\frac{\Delta\alpha}{\delta}, \frac{\Delta\beta}{\delta})$ as the *relative flow vector*, and a field of these vectors as the *relative flow field*. Since only the "left" quantities appear in the remainder of this work, we drop the "*l*" subscript to denote this.

An interesting point derived in [30] is that the x-component of the relative flow vector, given by equation (19), is identical to the ratio of the rate of change in the disparity to the disparity We now derive this relationship.

We know that the disparity, δ, is given in equation **(6)** as

$$\delta = \frac{b}{Z} \tag{21}$$

Hence, the rate of change of disparity, $\dot{\delta}$, is given as

$$\dot{\delta} = -\frac{b}{Z^2}\dot{Z} = -\delta\frac{\dot{Z}}{Z} \tag{22}$$

Referring to equation (3), we have

$$\frac{\dot{Z}}{Z} = \frac{1}{Z}(-\Omega_Y X + \Omega_X Y + T_Z) = -\Omega_Y x + \Omega_X y + T_Z/Z \tag{23}$$

Substituting this into equation (22), we find

$$\dot{\delta} = [\Omega_Y x - \Omega_X y - T_Z/Z]\delta \tag{24}$$

or

$$\frac{\dot{\delta}}{\delta} = \Omega_Y x - \Omega_X y - T_Z/Z. \qquad Q.E.D \tag{25}$$

Hence, we can obtain the relative flow field data in one of two ways, i.e.,

- take the relative flow between the two optic flow fields corresponding to the left and right cameras, or
- approximate the differential of the disparity field over the interval between the two time instances.

We choose to obtain the relative flow fields using the former method. This view is supported in in a psychophysical context since experimental studies [22] have shown some subjects to have areas of the visual field that are *blind* to static disparity and yet possess normal sensitivity to motion in depth. Hence, we believe that the extraction of motion in depth by computing the relative optic flow between a stereo pair is a closer model of the human perception of such motion.

4. Computation of the MID parameters using Stereoscopic Motion Constraints.

A Brief Introduction to the Approach

We employ a global technique to extract the three MID parameters $\{\Omega_X, \Omega_Y, T_Z\}$ in the presence of general motion of the camera and several independently moving objects.

The theoretical basis for this approach to the integration of the stereo and motion information was given in Section 3. Briefly, if u is the relative optic flow at image points in the left and right cameras corresponding to a single world point and δ the disparity between the left and right images at any one time instant for the same point, it was shown that the ratio of u to δ is a linear function of the image coordinates. The coefficients of this linear functional are the rotations about the X and Y axes and the translation along the Z axis, which are precisely the MID parameters being computed in our approach.

Algorithm Description

The inputs to the algorithm are the two flow fields, one each for the left and the right images, and the disparity field between the left and the right images at any one time instant.

The algorithm proceeds in the following three steps :

- Extract the relative flow field between the left and right images using the difference of the two optic flow fields along with the disparity field.

- Employ Adiv's technique, [1,2], for obtaining the segmentation of the monocular optic flow field corresponding to the left camera. Thus, we perform a segmentation of the optic flow field from pure motion information on the 2-D image plane in order to obtain a grouping of the vectors, where each segment corresponds to the motion of a roughly planar surface. We discuss the effect of performing a similar grouping on the relative flow fields in Section 4.1.

- Merge the segments on the 2D image plane (obtained from the previous segmentation step) based on a least squares minimization to compute the MID parameters for each of the merged regions. The output at this stage is a grouping of the image into regions that correspond to the same set (within some normalized value of the deviation) of three MID parameters, i.e., the two parameters of rotation in depth and the scaled translation in depth.

In the interest of robustness, the grouping obtained in the segmentation step is used as a template to guide the areas in the image where the minimization is employed. The actual optimiza·ion constraint is applied to the information in the relative flow field within each of a set of single or possibly merged group of segments in order to extract the MID parameters for them.

Also, since we deal only with MID parameters, the merged groups of regions *cannot* be interpreted as representing objects in the scene. They represent those regions in the image that have the same set (within some normalized value of the deviation) set of MID parameters. See Expt.4 (Figure 5e) in Section 5 for an example of a grouping wherein the background plane and a stationary ellipsoid in the environment get merged as one region simple because both have the same MID parameters relative to the camera.

We describe these two steps of the algorithm in the following two subsections.

4.1 Segmentation

At this stage of the algorithm, the optic flow field for the left camera is used to obtain a 2-D grouping of those flow vectors that are consistent with the motion of an approximately planar patch [1,2]. We also discuss the effects of performing a similar segmentation on the *relative flow field*.

Formulation Of The Segmentation Constraint

Let us first examine the use of optic flow for segmentation. We briefly review the segmentation process as developed in [1,2]. The viewer is advised to see these for more details. If we consider the flow field that is induced by the motion of a rigid planar surface described by

$$k_x X + k_y Y + k_z Z = 1. \qquad (26)$$

Then we can rewrite equations (4,5) as

$$\alpha = a_1 + a_2 x + a_3 y + a_7 x^2 + a_8 x y, \qquad (27)$$

$$\beta = a_4 + a_5 x + a_6 y + a_7 x y + a_8 y^2, \qquad (28)$$

where $\{a_1, \ldots, a_8\}$ are functions of the 3-D motion parameters, (T, Ω), of the objects in the environment (or conversely, the camera) and the three planar surface parameters, (k_x, k_y, k_z) :

$$\begin{aligned}
a_1 &= \Omega_Y + k_z T_X, \\
a_2 &= k_x T_X - k_z T_Z, \\
a_3 &= -\Omega_Z + k_y T_X, \\
a_4 &= -\Omega_X + k_z T_Y, \\
a_5 &= \Omega_Z + k_x T_Y, \\
a_6 &= k_y T_Y - k_z T_Z, \\
a_7 &= \Omega_Y - k_x T_Z, \\
a_8 &= -\Omega_X - k_y T_Z. \qquad (29)
\end{aligned}$$

The desired output is an organization of the optic flow field into *segments*, where each segment corresponds to the motion of a roughly planar surface. The constraint used to perform this grouping is that each segment thus formed be consistent with a single Ψ-transformation {equations (27,28)}. The Ψ-transformation space corresponds to the coefficients of the second order flow equations as functions of image coordinates. This space is 8-dimensional, and searching it for the Ψ-transformation that is consistent with the motion in the flow field would be very expensive computationally. Hence, the consistency constraint is approximately implemented by a two-step process :

- As a preprocessing step, the flow vectors are first grouped based on consistency with the 6-parameter linear approximation to equations 27,28 (an affine transformation) using a modified Hough transform. This yields *components*.

- These components are then merged together based on the minimization of an error measure derived using the full optimal Ψ-transformation.

We note several interesting features of this stage. The six parameter affine transformation corresponds to a Ψ-transformation with $a_7 = a_8 = 0$. In addition, the flow components $\alpha(\beta)$ depends only on the parameters $\{a_1, a_2, a_3\}(\{a_4, a_5, a_6\})$, i.e., the two components are decoupled. Because of this, the significant computational cost of a Hough transform on the 6-dimensional affine space can be mitigated by instead performing a seperate Hough transform on each of the two three-parameter affine spaces $\{a_1, a_2, a_3\}$ and $\{a_4, a_5, a_6\}$. In addition the transform is implemented on the 3-parameter affine spaces in a multiresolution scheme that makes use of the concept of dynamically quantized spaces wherein the transform is iteratively employed around the value estimated in the previous iteration using a finer resolution.

It is important to note that this segmentation step uses an approximation to determine the grouping. Hence, the values of the Ψ-transformation themselves are not used to compute the motion parameters. This grouping is used as a mask on the *relative flow field* in order to determine the values of the MID parameters without making any surface approximations, in order to increase the robustness of the algorithm.

Difficulties with direct use of the relative flow field for segmentation

We now discuss our reasons for using the pure motion flow fields, rather than the relative flow field, for segmentation purposes. Using the planar patch approximation (26) in the relative flow equations (19,20), we find

$$\frac{\Delta\alpha}{\delta} = b_0 + b_1 x + b_2 y. \quad (30)$$

$$\frac{\Delta\beta}{\delta} = 0, \quad (31)$$

where

$$\begin{aligned} b_0 &= -T_Z k_z, \\ b_1 &= \Omega_Y - k_x T_Z, \\ b_2 &= -\Omega_X - k_y T_Z. \end{aligned} \quad (32)$$

Upon comparing equations (29) and (32) we see that the second order components of the *optic* flow field, i.e., the coefficients a_7 and a_8, are precisely the first order components of the *relative* flow field, namely, b_1 and b_2 respectively:

$$\begin{aligned} a_7 &= b_1. \\ a_8 &= b_2 \end{aligned} \quad (33)$$

Grouping using the relative flow constraint would thus result in a set of relative flow vectors that are consistent with the same set of three parameters, b_0, b_1 and b_2 which amounts to dealing only with the second order components of the optic flow field, and disregarding the first order components. This creates incorrect grouping since in situations where the motion is such that the second order components of the flow field are very small, the relative flow field is going to be a function of very small first order coefficients. This occurs in situations where the translation and rotations in depth are small. A Hough transform on such a space will create false peaks in the parameter space and thus be unreliable. Thus for purposes of grouping, we would like to utilise all the available information and obtain as reliable a grouping as possible.

Hence, we use the optic flow (corresponding to the left camera) in order to obtain the segmentation mask to be used in the optimization on the relative flow field. In the current implementation the optic flow corresponding to the left camera is chosen to obtain the segmentation.

4.2 Optimization

Segments that are consistent with the same set of three MID parameters are merged together in this stage, which proceeds in several steps:

- As an initial step, optimal MID parameters and a related error measure (see 35) are computed for each of the segments obtained from the segmentation step. The MID parameters are computed using a least squares error minimization (see 34).
- Sets of segments are sequentially tested for merger by deciding if the relative flow field in them corresponds to a single set of MID parameters.
- The merging decisions are based on the degree of consistency of the relative flow vectors in the entire set of segments being tested for merging to the same set of three optimal MID parameters. This is done by comparing the error measure obtained by taking the entire set of segments with the error measures for each of the segments taken singly. Both the error measures are computed with respect to the three optimal MID parameters that are obtained for the merged set using a least squares minimization.
- Only segments that have not been included in any previous merging are included in the next merging
- The MID parameters are computed for the merged sets of segments.
- All segments that correspond to the stationary environment are grouped together as one merged segment.
- Independently moving objects are picked out, unless the flow corresponding to the regions in and around the objects is such as to produce ambiguities during the interpretation process.
- The computation at this stage is *linear* in complexity since the technique considers only the *neighbouring* segments for merging decisions. This is reasonable since we are searching for groups in the image that correspond to the same set of MID parameters rather than recognise object masks.

Minimization Process

It is required that the MID parameters be extracted in the simplest possible manner that preserves robustness in the entire computation. In general, it is possible to obtain them for each segment from the values of the relative flow field at three points in that segment. We could then think of ways to merge neighbouring segments that exhibit the same MID parameters. But two factors need to be considered First, the flow fields and the disparity field are generally prone to corruption by noise. Second, the current method of obtaining the relative flow field by taking the difference of two flow fields adds numerical errors to the data. Given these two sources of data distortion, it is important to use all possible information in computing the parameters. A least squares error formulation is well suited for this purpose since it is more global in nature and robust in implementation.

Computation of the MID parameters

The least squares formulation minimizes the error between the actual relative flow field value at a point and that predicted by the MID parameters and the depth of the point. Hence, given a set of flow vectors, it selects the optimal set of three MID parameters that are consistent with the minimal deviation in the relative flow field predicted by them from the actual relative flow field.

Optimization Constraint

Based on equations (19, 20), the error function to be minimized over the set of relative flow vectors corresponding to a single segment or a possibly merged set of them is

$$E(\Omega_X, \Omega_Y, T_Z) = \sum_{i=1}^{n} W_i \left[\left(\frac{\Delta\alpha}{\delta} \right)_i - \Omega_Y x_i + \Omega_X y_i + T_Z Z_i \right]^2. \quad (34)$$

where $T = (T_X, T_Y, T_Z)$ is the translation vector, $\Omega = (\Omega_X, \Omega_Y, \Omega_Z)$ is the rotation vector and Z_i is the depth of the environmental point corresponding to the image point (x_i, y_i). W_i is the weight (or confidence measure [4,5]) associated with the flow vector at the point (x_i, y_i). It is required to obtain the three MID parameters, $(\Omega_X, \Omega_Y, T_Z)$, that minimize the above functional. This is done by differentiating the functional with respect to each of them, and setting the resulting differential equal to zero. We can then solve the resulting matrix equation for the three MID parameters. The computation at this step is fast since we only need to solve a 3×3 linear system, with some additional multiplication for the weighting process and summation over the set of relative flow vectors. For the purposes of the present formulation, we use the disparity field, δ, to provide the absolute depth information, Z_i. Thus, we obtain the absolute translation in depth, T_Z, and the two rotations in depth, Ω_X and Ω_Y.

Given a set of n flow vectors, let the solution to the minimization constraint be given by $P^* = (\Omega_X^*, \Omega_Y^*, T_Z^*)$. The normalized value of the deviation in the actual relative flow field from that predicted by the solution P^* can then be given by

$$\sigma = \sqrt{\frac{E(P^*)}{\sum_{i=1}^{n} W_i}}. \tag{35}$$

5. Simulation Results

In this section, we use stereoscopic motion to compute the MID parameters of the objects in the environment and of the camera. The results are shown for a set of simulations that were devised to cover the following categories of motion for rigid objects –

- translation in depth alone,
- rotation in depth alone,
- combined translation and rotation in depth, and
- general, independent object and camera motion.

The input data are the simulated flow fields corresponding to the left and right cameras as well as the simulated static disparity field between the two cameras for the first time instance. Note that the flow fields are dense and could only correspond to highly textured surfaces. In the first four experiments, we have shown the algorithm performance for *ideal* flow and disparity fields. In the fifth and sixth experiments, we show the algorithm performance with gaussian noise added to the flow fields. Note that the algorithm assumes as input, the optic flow and disparity values, and hence bypasses the correspondence problem.

Values of the translation parameters and the surface parameters are in focal units, the flow and disparity vectors are in pixel units, and the rotation parameters are in radians. The image is 128×128. The field of view of each of the cameras is 45°. The baseline is 0.5 focal units, giving rise to disparity values of about 4 to 6 pixels for the simulated environment. The size and position of the objects are given in focal units and are with respect to the left camera, coordinate system, as are the motion values.

Experiment 1 : Object Translating in Depth

This experiment demonstrates the chief advantage of using stereoscopic constraints in motion computation, i.e., the *fast* computation of the motion of objects translating in depth. Such motion represents the direct motion of the object along a line parallel to the line of sight.

The simulated input motion and the computed translation in depth for the moving object are shown in Table 1. The environment consists of two distinct surfaces :

1. a plane described by the equation $Z = 100$.

size	*position*	*Object Translation (focal units)*	
		Input	*Computed*
2,2,2	-3,-1,15	$T_Z = 1.00$	$T_Z^{comp} = 1.08$

Table 1: Translation in depth of sphere of radius=2. Camera is stationary.

2. a sphere of radius=2, at position = (−3, −1, 15) translating with $T_Z = 1.00$.

The camera system is stationary.

Note that the computed value of the translation in depth is the absolute (not relative) value since disparity information is used to obtain the depth (see Table 1).

Experiment 2 : Object rotating in depth.

In this experiment, we demonstrate the ability of the model to find the rotation in depth of objects. This will be used in future studies to model the perception of the structure of rotating objects [9,28,29]. The motion of interest is a spinning motion about axes parallel to the image planes.

The simulated input motion and the computed rotation in depth for the moving object are shown in Table 2. The environment consists of two distinct surfaces :

1. a plane described by the equation $Z = 100$.
2. a sphere of radius=2, at position = (−3, −1, 15) rotating with $\Omega_X = 0.05$ and $\Omega_Y = 0.05$.

The value of the rotation in depth parameters computed by the algorithm are shown in Table 2.

size	*position*	*Object Rotation (radians)*	
		Input	*Computed*
2,2,2	-3,-1,15	$\Omega_X = 0.05$ $\Omega_Y = 0.05$	$\Omega_X^{comp} = 0.08$ $\Omega_Y^{comp} = 0.04$

Table 2: Rotation in depth of sphere of radius = 2. Camera is stationary.

Experiment 3 : General motion in depth of object

It is shown here that the algorithm makes no assumptions about the motion of the object(s) and is applicable in the presence of both rotation and translation in depth. This example is a simulation of the motion of a spinning ball being thrown towards the observer. The object motion consists of translational components along the the X and Z axes, and a single rotational component about the X axis (both stated with respect to the left camera coordinate system). The camera system is stationary.

The values of both the translation in depth and the rotations in depth are computed and are shown in Tabie 3.

The input being simulated is as follows. The environment consists of two distinct surfaces :

1. a plane described by the equation $Z = 100$.

size	*po. ition*	*Object Translation (focal units)*		*Object Rotation (radians)*	
		Input	*Computed*	*Input*	*Computed*
2,2,2	-3,-1,15	$T_X = 0.50$ $T_Z = 1.20$	$T_Z^{comp} = 1.12$	$\Omega_X = 0.05$	$\Omega_X^{comp} = 0.08$

Table 3: Motion in depth of sphere of radius=2. Camera is stationary. Note that only the MID parameters are computed.

2. a sphere of radius=2, with position = (1, 4, 15), with translation ($T_X = 0.5, T_Z = 1.20$) and rotation ($\Omega_X = 0.05$).

Note in Table 3 that T_X is not given. This is because our algorithm computes only the MID parameters.

Experiment 4 : General camera motion and independent object motion

In this example, we show the performance on a simulation of both camera motion and independent object motion. The camera motion is completely general with all the components of translation and rotation being present. The object motion has NO MID componen'ts, but has the other three components of motion, i.e., translations along the X and Y axes, and rotation about the Z axis.

The input being simulated is as follows - The environment consists of two distinct surfaces :

- the background described by a plane, $Z = X + 0.5Y + 50$.
- a sphere with position = (9, 9, 30), radius = 2 and motion described by $(T_X, T_Y, T_Z) = (0.5, -0.5, 0.0)$ and $(\Omega_X, \Omega_Y, \Omega_Z) = (0.00, 0.00, -0.19)$.
- a *stationary* ellipsoid with position = (−3, −1, 20) and size = (2, 5, 2).

The motion of the camera is $(T_X, T_Y, T_Z) = (0.5, 0.5, 1.0)$ and $(\Omega_X, \Omega_Y, \Omega_Z) = (0.02, -0.02, 0.04)$.

The flow fields corresponding to the left and right cameras as well as the disparity field between them at the first time instance are shown in Figures 4a, 4b and 4c, respectively. Figure 4d represents the segmentation mask obtained from the left optic flow field to be used as a mask in the optimization stage.

Note that in Figure 4e, the minimization step of the algorithm results in the *stationary* ellipsoid getting merged with the background. This is because both have the same MID attributes relative to the moving camera. The independently moving sphere is still retained as a seperate region since it has *MID* attributes relative to the moving camera that are distinct from those of the stationary background. This is an example of the fact that the algorithm does not pick out object masks, but does pick out regions in the image with the same set of MID parameters.

We note that the computed values of the MID parameters for the independently moving sphere are inaccurate. We attribute this to the following fact that the number of relative flow vectors (19,20) in the mask corresponding to the segmentation of the sphere is small. When the values are computed using the least squares minimization process on this small set of vectors, errors can be expected.

See Table 4 for the computed MID parameters.

Experiment 5 : Object Translating in Depth - Noisy Optic Flow Fields

In this experiment, we show the performance of the algorithm for the computation of the translation in depth of the object with *noisy optic flow fields* as input.

The environment simulated is identical to that in Experiment 1, with the camera being stationary. We have added gaussian noise of $\sigma = 0.3$ to the optic flow fields for the left and the right camera. These are shown in Figures 5a and 5b respectively. The results of the segmentation on the left optic flow field is shown in Figure 5c. Table 5 shows the results of the computation of the translation in depth for the object.

Note that the error in the computation is not significantly greater than that shown in Table 1 for experiment 1.

Experiment 6 : General camera motion and independent object motion - Noisy Optic Flow Fields

In this example, we show the performance of the algorithm in the presence of both camera motion and independent object motion with *noisy optic flow fields* as input.

The camera motion is completely general with all the components of translation and rotation being present. The object motion has NO MID components, but has the other three components of motion, i.e., translations along the X and Y axes, and rotation about the Z axis.

Note that the simulated input motion and environment description is identical to that in Experiment 4. However, gaussian noise of $\sigma = 0.3$ is added to the corresponding optic flow fields for the left and the right cameras. These are shown in Figures 6a and 6b. Figure 6c shows the results of segmenting the left optic flow field using Adiv's technique [1]. In Table 6, we show the MID parameters computed for the camera and the independently moving sphere.

Note that in Figure 6d, the *stationary* ellipsoid is still getting merged with the background, while the independently moving sphere continues to be picked out, just as in Experiment 4 (see Figure 4e).

The MID parameters computed for the independently moving sphere are even more inaccurate than those computed for the ideal flow fields in Exp.4. This we attribute to the fact that the number of relative flow vectors (19,20) in the mask corresponding to the segmentation of the sphere is small as well as the fact that the error in the relative flow vectors increases with non-ideal flow fields. We note that the computed values of the

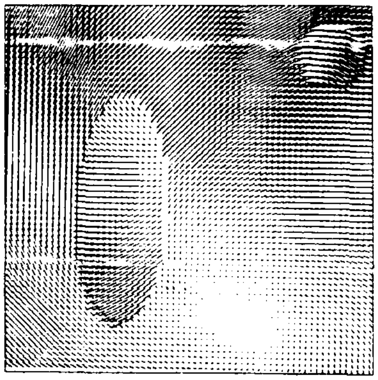


Figure 4a: Simulated *ideal*, dense optic flow field for the left camera.

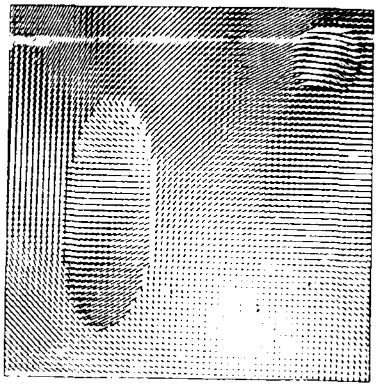


Figure 4b: Simulated *ideal*, dense optic flow field for the right camera.

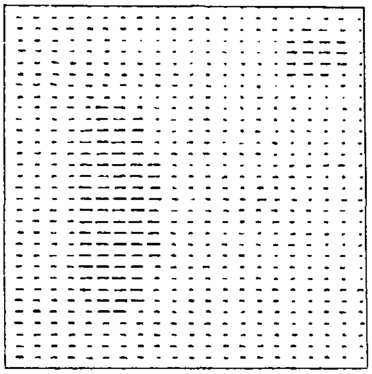


Figure 4c: Simulated *ideal* dense field of disparity vectors. Baseline is 0.5 focal units.

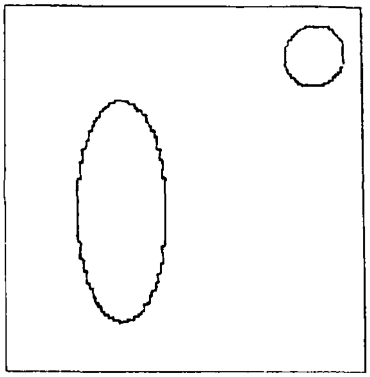


Figure 4d: Result of seg nentation performed using Adiv's algorithm [1].

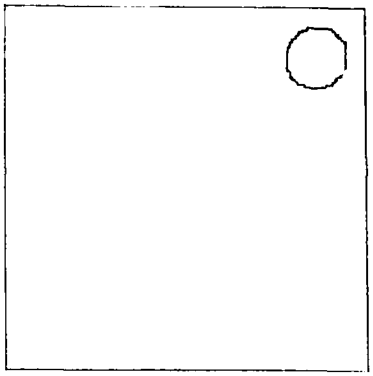


Figure 4e: Result of merger in the optimization step of the algorithm. Note that the independently moving sphere is picked out.

<table>
<tr><td rowspan="3"><i>sphere</i></td><td><i>size</i></td><td><i>position</i></td><td colspan="2"><i>Object Translation
(focal units)</i></td><td colspan="2"><i>Object Rotation
(radians)</i></td></tr>
<tr><td rowspan="2">2,2,2</td><td rowspan="2">9,9,30</td><td><i>Input</i></td><td><i>Computed</i></td><td><i>Input</i></td><td><i>Computed</i></td></tr>
<tr><td>$T_X = 0.50$
$T_Y = -0.5$
$T_Z = 0.00$</td><td>$T_Z^{comp} = 0.11$</td><td>$\Omega_X = 0.00$
$\Omega_Y = 0.00$
$\Omega_Z = -0.19$</td><td>$\Omega_X^{comp} = 0.04$
$\Omega_Y^{comp} = 0.02$</td></tr>
<tr><td></td><td><i>size</i></td><td><i>position</i></td><td colspan="2"><i>Object Translation
(focal units)</i></td><td colspan="2"><i>Object Rotation
(radians)</i></td></tr>
<tr><td><i>ellipsoid</i></td><td>2,5,2</td><td>-3,-1,20</td><td colspan="4">stationary</td></tr>
<tr><td><i>plane</i></td><td colspan="2">Z = X + 0.5Y + 50</td><td colspan="4">stationary</td></tr>
<tr><td rowspan="3"><i>camera</i></td><td colspan="3"><i>Camera Translation
(focal units)</i></td><td colspan="3"><i>Camera Rotation
(radians)</i></td></tr>
<tr><td><i>Input</i></td><td colspan="2"><i>Computed</i></td><td><i>Input</i></td><td colspan="2"><i>Computed</i></td></tr>
<tr><td>$T_X = 0.50$
$T_Y = 0.05$
$T_Z = 1.0$</td><td colspan="2">$T_Z^{comp} = 1.2$</td><td>$\Omega_X = 0.02$
$\Omega_Y = 0.02$
$\Omega_Z = 0.04$</td><td colspan="2">$\Omega_X^{comp} = 0.03$
$\Omega_Y^{comp} = -0.01$</td></tr>
</table>

Table 4: General camera motion with independent object motion.

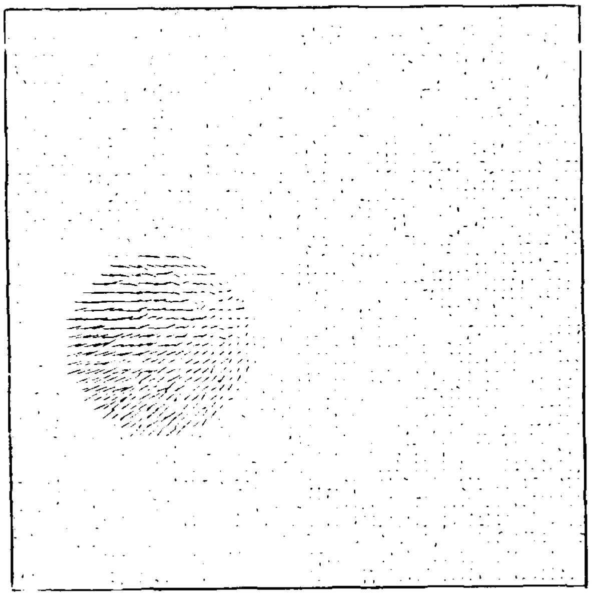


Figure 5a: Simulated *noisy*, dense optic flow field for the left camera.

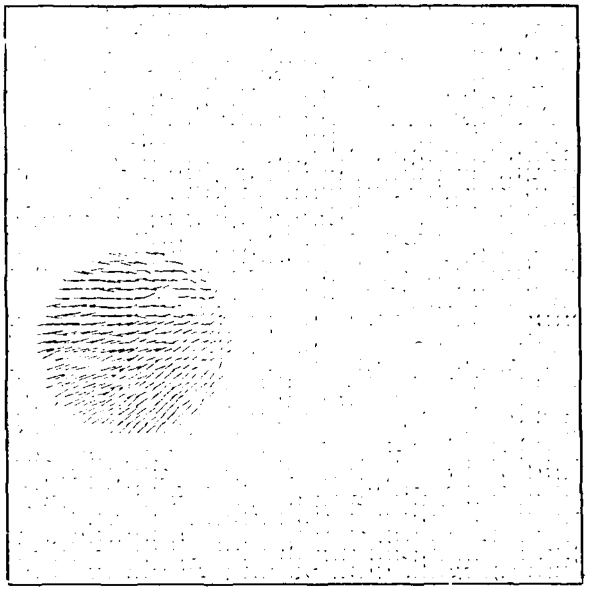


Figure 5b: Simulated *noisy*, dense optic flow field for the right camera.

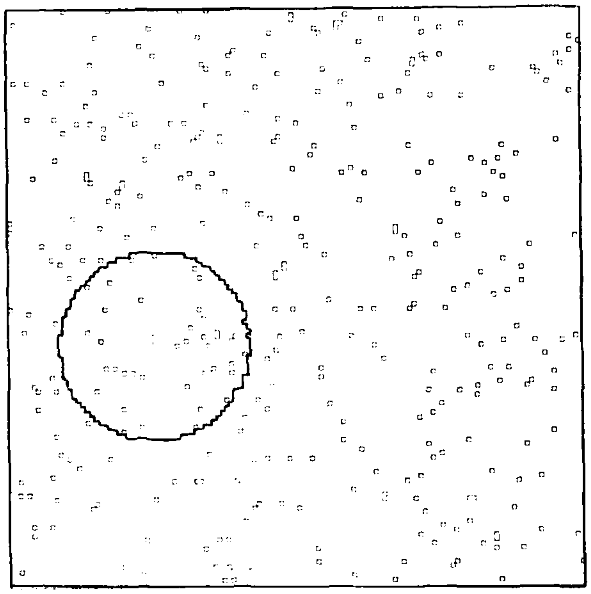


Figure 5c: Result of segmentation performed using Adiv's algorithm [1].

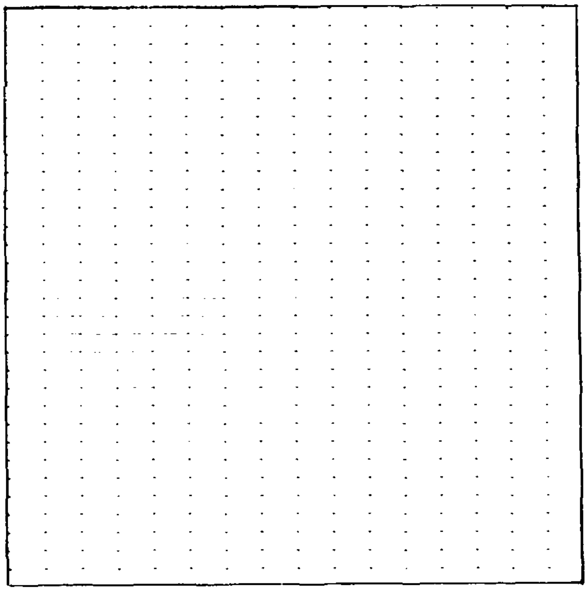


Figure 5d: Simulated *ideal* dense field of disparity vectors. Baseline is 0.5 focal units.

size	*position*	*Object Translation (focal units)*	
		Input	*Computed*
2,2,2	-3,-1.15	$T_Z = 1.00$	$T_Z^{comp} = 1.10$

Table 5: Translation in depth of sphere of radius = 2. Gaussian noise of $\sigma = 0.3$ added to the optic flow fields. Camera is stationary.

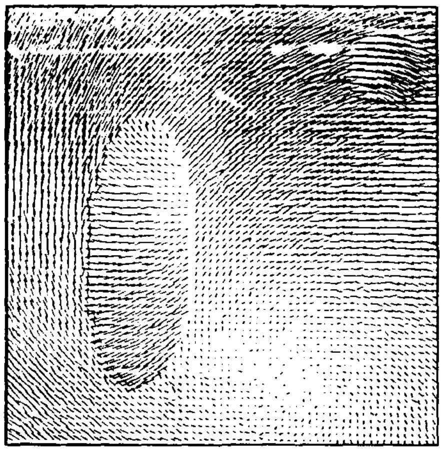


Figure 6a: Simulated *noisy*, dense optic flow field for the left camera.

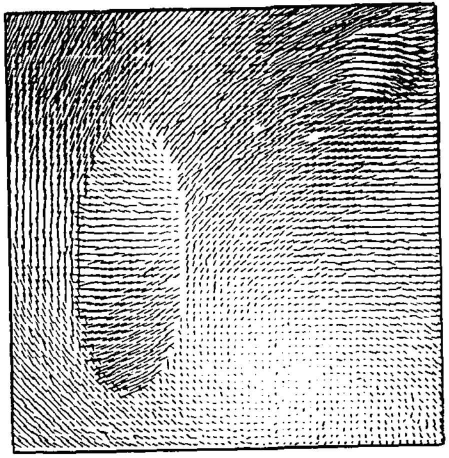


Figure 6b: Simulated *noisy*, dense optic flow field for the right camera.

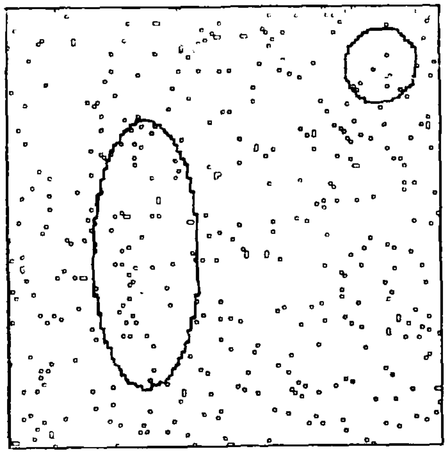


Figure 6c: Result of segmentation performed using Adiv's algorithm [1].

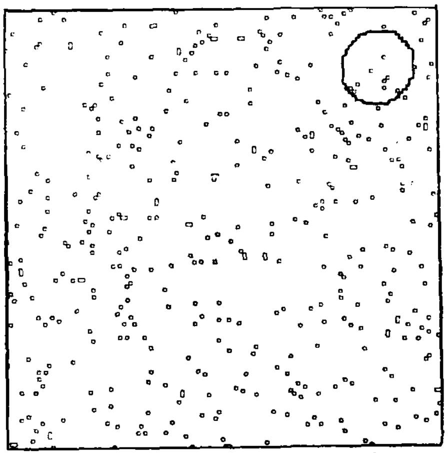


Figure 6d: Result of merger in the optimization step of the algorithm. Note that the independently moving sphere is still picked out while the stationary ellipsoid is merged with the background.

	size	*position*	*Object Translation (focal units)*		*Object Rotation (radians)*	
			Input	*Computed*	*Input*	*Computed*
sphere	2,2,2	9,9,30	$T_X = 0.50$ $T_Y = 0.5$ $T_Z = 0.00$	$T_Z^{comp} = 0.46$	$\Omega_X = 0.00$ $\Omega_Y = 0.00$ $\Omega_Z = -0.19$	$\Omega_X^{comp} = 0.08$ $\Omega_Y^{comp} = 0.06$
	size	*position*	*Object Translation (focal units)*		*Object Rotation (radians)*	
ellipsoid	2,5,2	-3,-1,20	stationary			
plane	Z = X + 0.5Y + 50		stationary			

	Camera Translation (focal units)		*Camera Rotation (radians)*	
	Input	*Computed*	*Input*	*Computed*
camera	$T_X = 0.50$ $T_Y = 0.05$ $T_Z = 1.0$	$T_Z^{comp} = 1.2$	$\Omega_X = 0.02$ $\Omega_Y = -0.02$ $\Omega_Z = 0.04$	$\Omega_X^{comp} = 0.03$ $\Omega_Y^{comp} = -0.01$

Table 6: General camera motion with independent object motion. Gaussian noise of $\sigma = 0.3$ added to the optic flow fields.

MID parameters for the camera are identical to those values computed for the ideal flow fields in Experiment 4 (see Table 4), demonstrating that the optimization step shows robust performance when the number of relative flow vectors available for the computation is not small, even if they are erroneous.

6. Summary

We have so far described how stereoscopic motion can be used to extract the MID parameters. The chief motivation was the need to obtain the MID parameters, particularly translation in depth, as quickly as possible. These parameters can be obtained from a general purpose motion algorithm such as Adiv's [1], which computes all the six motion parameters. But these algorithms are computationally intensive and do not permit a *quick yet reliable* grouping of the information from the imagery into regions corresponding to objects moving in depth. One of the chief reasons for this is that the computations need to deal with nonlinear equations that relate the flow information to the motion and structure parameters.

Use of stereoscopic motion for the computation of the MID parameters simplifies and speeds up the computation because of the following reasons -

1. The chief advantage provided by the integration of stereo and motion information for extracting motion in depth is that stereo information provides additional constraints that make it possible to formulate a *linear* relationship between the data in the flow and disparity fields and the motion in depth parameters, {see equations (19, 20)}. This, in turn, makes it possible to devise a direct computation without any hypothesizing of the motion parameters such as is done in [1,20].

2. The technique does *not* consider all possible subsets of the segments for grouping purposes. It only considers the *neighbouring* segments for merger decisions. This is reasonable since we are searching for groups in the image that correspond to the same three MID parameters, rather than seeking object masks. This makes the complexity of this step linear, rather than exponential (as in 1)

3. The computation of the three MID parameters are the result of a direct computation using the optimization constraint on the relative flow field. This can be used to directly compute the remaining three motion parameters, again, without employing a hypothesize and test scheme. Hence, extracting all the motion parameters becomes a problem of handling two sets of linear functionals.

4. The algorithm can be employed on motion sequences that include several independently moving objects and involve the general translation and rotation of the objects and the camera.

A problem that we hope to resolve in future work has to do with the nature of the relative flow fields. Since they are computed as a difference of the two flow fields, they are susceptible to error if the flow values are small. Hence, the *optimization* step is more accurate with larger motion values. Incorporating minimization norms that take this into account may help.

In the current implementation, we use the disparity field to provide us with the depth information in the computation. In future work, we hope to use the current results as a startup process and extend to a multi-frame paradigm for computing the motion over several frames as well as using this disparity information as a prediction of the depth and refining the depth map over several frames.

Acknowledgements

The authors would like to thank P. Anandan, G. Adiv, Debbi Strahman and Philip Kahn for many valuable discussions. Thanks are also due to Ed Riseman for his comments on the manuscript, and to the developers of the VISIONS system - in particular, Robert Heller - for always being helpful in ironing out implementation glitches. This research was supported by DARPA under grant N00014-82-K-0464. Also by DARPA and RADC under contract F30602-87-C- 0140.

References

[1] G. Adiv," Determining Three-Dimensional Motion and Structure from Optical Flow Generated by Several Moving Objects," *IEEE Trans. on Pattern Analysis and Machine Intelligence* Vol. PAMI-7, July 1985, 384-401.

[2] G. Adiv,"Interpreting Optical Flow," *Ph. D. Dissertation*, 1985, University Of Massachusetts, COINS TR 85-35.

[3] J. (Y). Aloimonos and I. Rigoutsos," Determining the 3-D motion of a rigid planar patch without correspondence, under perspective projection," *Proceedings of the IEEE Motion Workshop, Charleston, S.C.*, 1986, 167-174.

[4] P. Anandan, "Computing Dense Displacement Fields with Confidence Measures In Scenes Containing Occlusion," *SPIE Intelligent Robots and Computer Vision Conference* Vol. 521, 1984, 184-194. See also COINS TR 87-2, (UMass, Amherst).

[5] P. Anandan and R. Weiss, "Introducing A Smoothness Constraint In A Matching Approach For The Computation Of Optical Flow Fields," *IEEE Third Workshop On Computer Vision: Representation And Control* 1985, 186-194.

[6] P. Balasubramanyam, "Computation Of Motion-in-Depth Parameters Using Stereoscopic Motion Constraints " *IEEE Computer Society Workshop on Computer Vision* 1987, 349-351.

[7] J. Barron, "A Survey Of Approaches For Determining Optic Flow, Environmental Layout and Egomotion," University Of Toronto Technical Report No. RBCV-TR-84-5, 1984.

[8] K. I. Beverley and D. Regan, "Evidence for the Existence of Neural Mechanisms Selectively Sensitive to the Direction of Movement In Space," *Journal Of Physiology*, **235**, 1973, 17-29

[9] M. L. Braunstein, "Perception of Rotation in Depth: The Psychological Evidence," *ACM Workshop on Motion* 1983, 119-124.

[10] J. Q. Fang and T. S. Huang,"Estimating 3-D Movement of a Rigid Object: Experimental Results," *Proceedings of the International Joint Conference on Artificial Intelligence, Karlsruhe, Germany* 1983, 1035-1037

[11] B. K. P. Horn and E. J. Weldon, "Computationally Efficient Methods For Recovering Translational Motion," *Proceedings of the First International Conference On Computer Vision* June, 1987, 2-11

[12] T. S. Huang and S. D. Blostein, "Robust Methods for Motion Estimation based on Two Sequential Stereo Image Pairs," *Proceedings of the IEEE Motion Workshop, Bellaire, Mich.*, 1985, 518-523.

[13] M. R. M. Jenkin, "The Stereopsis of Time-Varying Images," *Technical Report*, University of Toronto, Toronto, Ontario, Canada, (RBCV-TR-84-3), Sept. 1984

[14] J. S. Lappin, J. F. Doner and B. L. Kottas, "Minimal Conditions for the Visual Detection of Structure and Motion in Three Dimensions," *Science*, 1980, **209**, 717-719.

[15] D. T. Lawton, "Processing Dynamic Image Sequences from a Moving Sensor," *Ph.D Dissertation* (TR 84-05), Computer and Information Science Dept., University Of Massachusetts, 1984

[16] H. C. Longuet-Higgins and K. Pradzny, "The Interpretation of a Moving Retinal Image," *Proceedings of the Royal Society of London*, July 1980, **B**(208):385-397.

[17] W. R. Miles, "Movement interpretations of the silhouette of a revolving fan," *American Journal of Psychology*, 1931, **43**, 392-405.

[18] K. M. Mutch, "Determining Object Translation Information using Stereoscopic Motion," *IEEE Transactions on Pattern Analysis and Machine Intelligence* Vol. PAMI-8, Nov. 1986, 750-755.

[19] J. T. Petersik, "Rotation Judgements and Depth Judgements: Seperate or Dependent Processes," *Perception and Psychophysics*, 1980, **27**, 588-590.

[20] K. Pradzny, "Determining the Instantaneous Direction of Motion from Optical Flow Generated by a Curvilinearly Moving Observer," *Proceedings of the IEEE Conference on Pattern Recognition and Image Processing, Dallas, Texas* 1981, 109-14.

[21] D. Regan and K. I. Beverley, "Binocular and Monocular Stimuli for Motion-In-Depth: Changing Disparity and Changing Size Inputs Feed the Same Motion-In-Depth Stage," *Vision Research*, 1979, **19**, 1331-1342

[22] D. Regan, "Visual Processing of Four Kinds of Relative Motion," *Vision Research*, 1986, **26**, 127-145

[23] W. Richards, "Structure from Motion and Stereo," *Journal Of the Optical Society of America* Feb. 1985, **2** 343-349.

[24] J. H. Rieger and D. T. Lawton, "Determining the Instantaneous Axis of Translation from Optic Flow Generated by Arbitrary Sensor Motion," *Proceedings of the Workshop on Motion: Representation and Perception, Toronto, Canada* 1983, 33-41.

[25] J. W. Roach and J. K. Aggarwal, "Determining the Movement Of Objects from a Sequence of Images," *IEEE Transactions on Pattern Analysis and Machine Intelligence* **2**, Nov. 1980, 554-562.

[26] W. B. Thompson, K. M. Mutch and V. A. Berzias, " Analyzing Object Motion Based on Optical Flow," *Proceedings of the International Conference on Pattern Recognition, Montreal, Canada* 1984, 791-794.

[27] R. Y. Tsai and T. S. Huang, "Uniqueness and Estimation of Three-Dimensional Motion Parameters of Rigid Objects with Curved Surfaces," *IEEE Transactions on Pattern Analysis and Machine Inteligence* **6** Jan. 1984, 13-27.

[28] S. Ullman, The Interpretation Of Visual Motion, MIT Press, 1979, Cambridge & London.

[29] S. Ullman, "Maximizing Rigidity: The Incremental Recovery Of 3-D Structure From Rigid And Rubbery Motion," *A.I.Memo No. 721*, MIT, June, 1983.

[30] A. M. Waxman and J. H. Duncan, "Binocular Image Flows: Steps toward stereo-motion fusion," *IEEE Transactions on Pattern Analysis and Machine Intelligence* Vol. PAMI-8, Nov.1986, 715-729.

[31] A. M. Waxman and K. Wohn, Image Flow Theory: A Framework for 3-D Inference from Time-Varying Imagery, *Chapter in Advances in Computer Vision (Erlbaum Publishers)*.

IS CORRESPONDENCE NECESSARY FOR THE PERCEPTION OF STRUCTURE FROM MOTION?

Eiki Ito
John (Yiannis) Aloimonos

Computer Vision Laboratory
Center for Automation Research
University of Maryland
College Park, MD 20742-3411

ABSTRACT

The fundamental assumption of almost all existing computational theories for the perception of structure from motion is that moving elements on the retina correspond projectively to identifiable moving points in three-dimensional space. Furthermore, these computational theories are based on the fundamental idea of retinal motion, i.e. they use as their input the velocity with which image points are moving (optic flow or discrete displacements). In this research, we investigate the possibility for the development of computational theories for the perception of structure from motion that are not based on the concept of the velocity of individual image elements, i.e. they do not use optic flow or displacements as input.

1. INTRODUCTION

The problem of structure from motion has received considerable attention lately (Ullman, 1979; Longuet-Higgins & Prazdny, 1980; Tsai and Huang, 1984). The problem is to recover the three-dimensional motion and structure of a moving object from a sequence of its images. Even though computation of structure and 3-D motion are equivalent when the retinal motion is given, the two problems have received different names, the former "Structure from Motion" and the latter "Passive Navigation". We will refer to them interchangeably.

Basically there have been two approaches toward solving this problem. The first assumes "small" motion. In this case, if the three-dimensional intensity function (one temporal and two spatial arguments) is locally well behaved and its spatio-temporal gradients are defined, then the image velocity field (or optic flow) may be computed (Horn and Schunck, 1981; Hildreth, 1984). The second approach assumes that the motion is "large" and measurement of image motion entails solving the correspondence problem. Imaged feature points due to the same three dimensional artifact (e.g. texture element, edge junction, etc.) in two successive dynamic frames are assumed to be identified correctly. Algorithms using this approach compute 3-D transformation parameters from the displacements field (Tsai and Huang, 1984; Longuet-Higgins, 1981; Ullman, 1979).

The fact is that computing retinal motion (optic flow or discrete displacements) is an ill-posed problem in the sense of Hadamard (1923); that means that in order to compute retinal motion we must make some assumptions about the world that might not always be present in the imaged scene. Until a rigorous theory of discontinuous regularization is developed, the problem of computing retinal motion remains a very hard one, as also acknowledged by Ullman (Ullman, 1981). Without arguing about whether or not correspondence is a basic process in the human visual system (and it should be so in a computer vision system), we investigate the problem of structure from motion without using correspondences. We develop computational theories for the problem that are not based on the concept of local retinal motion or correspondence of tokens. In this way, we show that in principle, correspondence or retinal motion are not necessary for computing "structure from motion". To understand whether or not algorithms based on correspondenceless "structure from motion" computational theories are used by the human visual system is beyond the scope of this paper.[1] Since our research interests lie primarily in machine vision, we will claim only that our correspondenceless computational theories may be used as the basis of algorithms for the problem of machine "passive navigation", because they are not based on the concept of retinal motion or correspondence, whose computations are ill-posed problems (Poggio et al., 1985).

The organization of the paper is as follows: Section 2 reviews previous work, and then we progressively build up our computational theories, by starting with some assumptions which are eliminated in the course of our analysis. So, Section 3 addresses the problem in the case where the depth of the scene is known, Section 4 solves the problem when the shape (something less than depth) is known, and Section 5 addresses the problem in the general case, where nothing is known about the scene. All treatments are done for perspective projection. Finally, Section 6 concludes the work and discusses future research.

2. PREVIOUS RESEARCH

Addressing the problem of structure from motion in a correspondenceless way is a relatively new idea. According to our knowledge, the problem was first addressed by Aloimonos and Brown (1984) in a simple way. It was shown in this work that the problem could be solved in the case of pure rotation.

The support of the Defense Advanced Research Projects Agency and the U.S. Army Center for Night Vision and Electro Optics under Contract DAAB07-86-K-F073 is gratefully acknowledged, as is the help of Barbara Bull in preparing this paper. Part of this work was done while Eiki Ito was visiting the Computer Vision Lab during 1987, on a Japanese Government fellowship.

[1] However, some researchers might claim so (Jenkin and Kolers, 1986).

In the case of egomotion, when there exists only pure rotation, the optic flow (u, v) at every image point (x, y) is given by

$$u(x,y) = \omega_1 xy - \omega_2(x^2+1) + \omega_3 y \qquad (1)$$

$$v(x,y) = \omega_1(y^2+1) - \omega_2 xy - \omega_3 x \ , \qquad (2)$$

where $\Omega = (\omega_1, \omega_2, \omega_3)$ is the rotation of the observer (see Figure 1).

On the other hand, if $s(x, y, t)$ is the time varying image intensity function, then the following relation is true (Horn, 1985):

$$s_x u + s_y v + s_t = 0 \qquad (3)$$

where the subscripts denote partial differentiation. Substituting (1) and (2) into (3), we get a linear equation in the unknowns $\omega_1, \omega_2, \omega_3$ whose coefficients are functions of retinal position and the image intensity function and its spatiotemporal derivatives. Considering this equation in many image points we get, through some aggregation method (least squares; Hough transform), the unique solution. Of course in this case structure cannot be computed, because there is no translational motion.

At about the same time Kanatani independently was devising general methods for recovering structure from motion without correspondence for planar surfaces, using the idea of linear features introduced in (Kanatani, 1984, 1985). At the same time Negahdaripour and Horn independently addressed the problem in a rigorous way and provided a uniqueness analysis for planar surfaces (Negahdaripour and Horn, 1985). Later, Horn and Weldon (1986) worked on the case of non-planar surfaces and pure translation and devised a method that searches for the focus of expansion and works under some assumptions. They also provided a robustness analysis for the case of pure rotation. In the meantime, independently Huang talked about correspondenceless detection of 3-D motion in the case of planes in a topical meeting at an Optical Society Conference (Huang, 1985) and Amari and Maruyama in Japan independently solved the problem for the case of planar surfaces (Amari and Maruyama, 1986). In the meantime Kanatani was improving his techniques for planar surfaces and for polyhedral ones under the assumption of knowledge of the structure (Kanatani and Chou, 1987).

This is, according to our knowledge, the sequence of research efforts up to the present.

3. KNOWN DEPTH

Here we treat the case of passive navigation when the depth of points in the scene is known. In the differential case the problem is trivial. Indeed, the optic flow (u, v) at every point is given by

$$u(x,y) = \frac{-U + xW}{Z} + \omega_1 xy - \omega_2(x^2+1) + \omega_3 y \qquad (4)$$

$$v(x,y) = \frac{-V + yW}{Z} + \omega_1(y^2+1) - \omega_2 xy - \omega_3 x \qquad (5)$$

where (U, V, W) is the translation and $\Omega = (\omega_1, \omega_2, \omega_3)$ the rotation of the camera, and Z the depth of the 3-D point whose image is the point (x, y). Substituting (4) and (5) into (3) we get a linear equation with unknowns $U, V, W, \omega_1, \omega_2$ and ω_3. Considering this equation at many points we get a solution in general, using a least squares or Hough transform methodology.

In the discrete case, the problem is more involved and we present here its solution. We consider a set $A = \{(X_i, Y_i, Z_i),\ i = 1, \ldots, n\}$ of 3-D points that undergoes a rigid transformation and becomes the set $A' = \{(X_i', Y_i', Z_i'),\ i = 1, \ldots, n\}$. We wish to recover the transformation (that transformed set A to set A') without considering any point to point correspondences. A part of the forthcoming discussion follows [Aloimonos and Rigoutsos, 1986].

Any rigid motion can be analyzed into a rotation plus a translation; the rotation axis can be considered as passing through any point in space, but after this point is chosen, everything else is fixed.

If we consider the rotation axis as passing through the origin of the coordinate system, then if the point

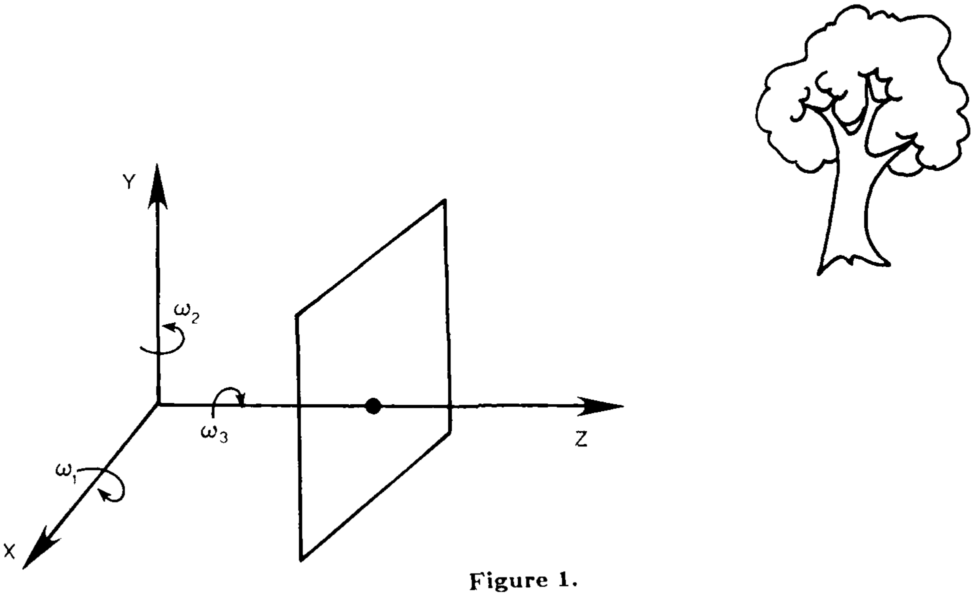


Figure 1.

$(X_i', Y_i', Z_i') \in A$ moves to a new position $(X_i', Y_i', Z_i') \in A'$, the following relation holds:

$$(X_i', Y_i', Z_i')^T = \mathbf{R}(X_i, Y_i, Z_i)^T + \mathbf{T} \qquad i = 1, 2, 3, \ldots, n \quad (6)$$

where $\mathbf{R}$ is the 3×3 rotation matrix and $\mathbf{T} \equiv (\Delta X, \Delta X, \Delta Z)^T$ is the translation vector. We wish to recover the parameters $\mathbf{R}$ and $\mathbf{T}$, without using any point-to-point correspondences.

Let

$$(X_i', Y_i', Z_i')' \equiv \mathbf{P}_i \text{ and } (X_i', Y_i', Z_i')' \equiv \mathbf{P}_i' \qquad i = 1, 2, 3, \ldots,$$

Then, equation (6) becomes

$$\mathbf{P}_i = \mathbf{R}\mathbf{P}_i' + \mathbf{T} \qquad i = 1, 2, 3, \ldots, n$$

Summing up the above n equations and dividing by the total number of points, n, we get

$$\frac{\sum_{i=1}^{n} \mathbf{P}_i}{n} = \mathbf{R} \frac{\sum_{i=1}^{n} \mathbf{P}_i}{n} + \mathbf{T} \qquad (7)$$

From (7) it is clear that if the rotation matrix $\mathbf{R}$ is known, then the translation vector T can be computed. So, in the sequel, we will describe how to recover the rotation matrix $\mathbf{R}$. In order to get rid of the translational part of the motion we shall transform the 3-D points to "free" vectors by subtracting the center-of-mass vector.

Let, therefore, $\mathbf{CM_A}$ and $\mathbf{CM_A'}$ be the center-of-mass vectors of the sets of points A and A' respectively; i.e. $\mathbf{CM_A} = \sum(\mathbf{P}_i / n)$ and $\mathbf{CM_A'} = \sum(\mathbf{P}_i' / n)$. We furthermore define

$$\mathbf{v}_i = \mathbf{P}_i - \mathbf{CM_A} \qquad i = 1, 2, 3, \ldots, n$$

$$\mathbf{v}_i' = \mathbf{P}_i' - \mathbf{CM_A'} \qquad i = 1, 2, 3, \ldots, n$$

With these definitions, the motion equation (6) becomes

$$\mathbf{v}_i' = \mathbf{R}\,\mathbf{v}_i \qquad i = 1, 2, 3, \ldots, n$$

where $\mathbf{R}$ is the **(orthogonal)** rotation matrix.

If we know the correspondences of some points (at least three) then the matrix $\mathbf{R}$ can in principle be recovered, and such efforts have been published. But we would like to recover matrix $\mathbf{R}$ without using any point correspondences. Let

$$\mathbf{v}_i = (v_{x_i}, v_{y_i}, v_{z_i}) \qquad i = 1, 2, 3, \ldots, n$$

$$\mathbf{v}_i' = (v_{x_i}', v_{y_i}', v_{z_i}') \qquad i = 1, 2, 3, \ldots, n$$

Note that $\mathbf{v}_i$ and $\mathbf{v}_i'$ are the position vectors of the members of sets A and A' respectively with respect to their center-of-mass coordinate systems.

We wish to find a quantity that will uniquely characterize the sets A and A' in terms of their "relationship" (rigid motion transformation). We have found that the matrix consisting of the second order moments of the vectors $\mathbf{v}_i$ and $\mathbf{v}_i'$ has these properties. In particular, let

$$\mathbf{V} \equiv \begin{bmatrix} \sum_{i=1}^{n} v_{x_i}^2 & \sum_{i=1}^{n} v_{x_i} v_{y_i} & \sum_{i=1}^{n} v_{x_i} v_{z_i} \\ \sum_{i=1}^{n} v_{y_i} v_{x_i} & \sum_{i=1}^{n} v_{y_i}^2 & \sum_{i=1}^{n} v_{y_i} v_{z_i} \\ \sum_{i=1}^{n} v_{x_i} v_{z_i} & \sum_{i=1}^{n} v_{y_i} v_{z_i} & \sum_{i=1}^{n} v_{z_i}^2 \end{bmatrix}$$

$$\mathbf{V}' \equiv \begin{bmatrix} \sum_{i=1}^{n} v_{x_i}'^2 & \sum_{i=1}^{n} v_{x_i}' v_{y_i}' & \sum_{i=1}^{n} v_{z_i}' v_{x_i}' \\ \sum_{i=1}^{n} v_{y_i}' v_{x_i}' & \sum_{i=1}^{n} v_{y_i}'^2 & \sum_{i=1}^{n} v_{y_i}' v_{z_i}' \\ \sum_{i=1}^{n} v_{x_i}' v_{z_i}' & \sum_{i=1}^{n} v_{y_i}' v_{z_i}' & \sum_{i=1}^{n} v_{z_i}'^2 \end{bmatrix}$$

From these relations, we have

$$\mathbf{V}' = \sum_{i=1}^{n} (v_{x_i}'\, v_{y_i}'\, v_{z_i}')' (v_{x_i}'\, v_{y_i}'\, v_{z_i}') =$$

$$= \sum_{i=1}^{n} \mathbf{R}(v_{x_i}, v_{y_i}, v_{z_i})' (v_{x_i}, v_{y_i}, v_{z_i}) \mathbf{R}' = \mathbf{R}\mathbf{V}\mathbf{R}'$$

So,

$$\mathbf{V}' = \mathbf{R}\,\mathbf{V}\,\mathbf{R}' \qquad (8)$$

At this point it should be mentioned that equation (8) represents an invariance between the two sets of 3-D points A and A', since the matrices V and V' are similar. In other words we have discovered that matrix V remains invariant under rigid motion transformation. From now on, the recovery of the rotation matrix $\mathbf{R}$ is simple and comes from basic linear algebra. Furthermore, equation (8) implies that the matrices V and V' have the same set of eigenvalues [Stewart, 1980].

But since V and V' are symmetric matrices, they can be expanded in their eigenvalue decomposition, i.e. there exist matrices S, T such that

$$V = SDS^t \qquad (9)$$

$$V' = TDT^t \qquad (10)$$

where S, T are orthogonal matrices having as columns the eigenvectors of the matrices V and V' respectively (e.g. the i^{th} column corresponding to the i^{th} eigenvalue) and D is a diagonal matrix consisting of the eigenvalues of the matrices V and V'. We have to mention at this point that in order to make the decomposition unique we require that the eigenvectors in the columns of matrices S and T be orthonormal.

From equations (8), (9) and (10) we derive that matrices T and $\mathbf{R}S$ both consist of the orthonormal eigenvectors of matrix V'. In other words, the columns of matrices $\mathbf{R}S$ and T must be the same, with a possible change of sign. So, the matrix $\mathbf{R}S$ is equal to one of eight possible matrices $T_i, i = 1, \ldots, 8$. Thus, $\mathbf{R} = T_i S^T$, $i = 1, \ldots, 8$. But the rotation matrix is orthogonal and it has determinant equal to one. Furthermore, if we apply matrix $\mathbf{R}$ to the set of vectors $\mathbf{v}_i$ then we should get the set of vectors $\mathbf{v}_i'$. So, given the above three conditions and Chasles' theorem, the matrix $\mathbf{R}$ can be computed uniquely. Thus, we can compute 3-D *motion without point correspondences.* The robustness against noise in this method (and noise arises from several sources) is studied elsewhere. Here, we wanted to theoretically demonstrate the feasibility of the approach.

4. KNOWN SHAPE

Here we treat the egomotion problem when the shape of the object (or scene) in view is known. The formulation and treatment here follow Ito and Aloimonos (1987). Assume that a surface W, whose shape is known, is moving rigidly while its images are taken (Figure 2).

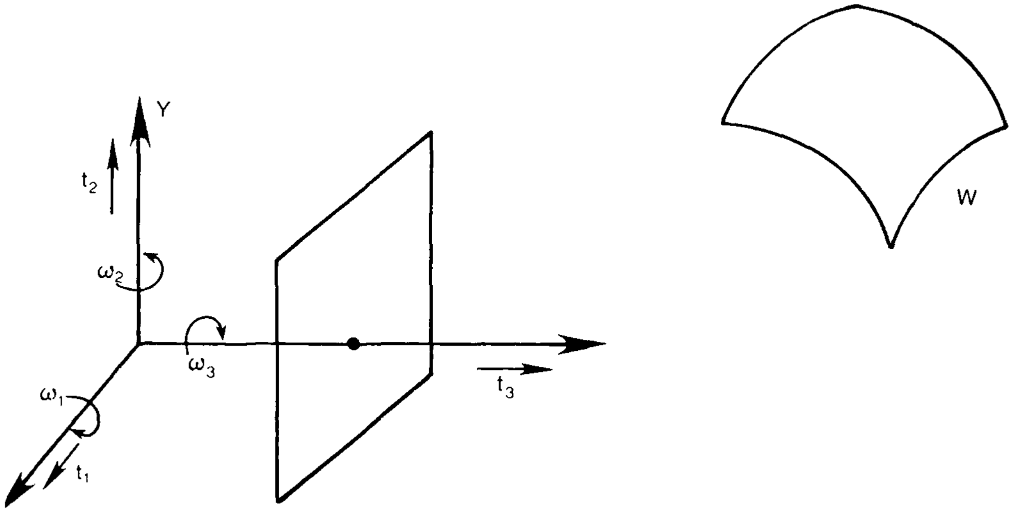


Figure 2.

Let the motion of the surface consist of a translation $T=(t_1, t_2, t_3)$ and a rotation $\Omega=(\omega_1, \omega_2, \omega_3)$. So, if $\mathbf{X}=(X, Y, Z)$ is a point on the surface W, then its velocity $\mathbf{V}(\mathbf{X})=(V_1, V_2, V_3)=T+\Omega\times\mathbf{X}$. This velocity can be written as $\mathbf{V}(\mathbf{X})=\sum_{k=1}^{6} m_k\mathbf{V}_k(\mathbf{X})$, where m_k are the motion parameters and $\mathbf{V}_k$ are fixed vector fields. Indeed

$$\begin{aligned}
m_1&=t_1 & \mathbf{V}_1(\mathbf{X}) &=(1\,0\,0)^T\\
m_2&=t_2 & \mathbf{V}_2(\mathbf{X}) &=(0\,1\,0)^T\\
m_3&=t_3 & \mathbf{V}_3(\mathbf{X}) &=(0\,0\,1)^T\\
m_4&=\omega_1 & \mathbf{V}_4(\mathbf{X}) &=(0\,{-Z}\;Y)^T\\
m_5&=\omega_2 & \mathbf{V}_5(\mathbf{X}) &=(Z\;0\;{-X})^T\\
m_6&=\omega_3 & \mathbf{V}_6(\mathbf{X}) &=(-Y\;X\;0)^T
\end{aligned}$$

Let $\mathbf{P}$ denote the perspective projection (from the object surface to the image plane), i.e. if $\mathbf{x}=(x, y)$ is the image of point $\mathbf{X}=(X, Y, Z)\in W$, then

$$\mathbf{P}(\mathbf{X})=\left(\mathbf{P}_1(\mathbf{X}), \mathbf{P}_2(\mathbf{X})\right)=\left(\frac{fX}{Z}, \frac{fY}{Z}\right)=(x, y)=\mathbf{x}$$

where f is the focal length of the camera, assumed here to be unity. Since the shape of the object in view is assumed to be known (let it be the function h), a mapping $\mathbf{P}_{h^{-1}}$: (image plane) → (object surface) is defined such that $\mathbf{P}\left(\mathbf{P}_{h^{-1}}(\mathbf{x})\right)=\mathbf{x}$. The optic flow at point $\mathbf{x}$ is given by

$$\dot{\mathbf{x}}=(\dot{x}, \dot{y})=\frac{d}{dt}\mathbf{P}(\mathbf{X})=\frac{\partial \mathbf{P}}{\partial \mathbf{X}}\mathbf{V}(\mathbf{X}).$$

Clearly $\frac{\partial \mathbf{P}}{\partial \mathbf{X}}$ and $\mathbf{V}(\mathbf{X})$ are functions of $\mathbf{X}$, but using the inverse projection (from the image to the surface), we can consider them as functions of $\mathbf{x}$ (retinal coordinates).

So, $\frac{\partial \mathbf{P}}{\partial \mathbf{X}}=\frac{\partial \mathbf{P}}{\partial \mathbf{X}}\left(\mathbf{P}_{h^{-1}}(\mathbf{x})\right)$ and $\mathbf{V}=\mathbf{V}\left(P_{h^{-1}}(\mathbf{x})\right)$ But since $\mathbf{V}=\sum_{k=1}^{n} m_k\,\mathbf{V}_k$, we have

$$\dot{\mathbf{x}}=\sum_{k=1}^{n} m_k\,\mathbf{\Phi}_k(\mathbf{x}) \tag{11}$$

with

$$\mathbf{\Phi}_k(\mathbf{x})=\frac{\partial \mathbf{P}}{\partial \mathbf{X}}\left(\mathbf{P}_{h^{-1}}(\mathbf{x})\right)\mathbf{V}_k\left(P_{h^{-1}}(\mathbf{x})\right)$$

It has to be emphasized that $\mathbf{\Phi}_k$ are functions of the shape, and so are known.

4.1. Linear features

A linear feature l is a linear functional over the image $S(x, y)$, i.e.

$$l=\iint_D S(x, y)\,\mu(x, y)\,dxdy$$

where $\mu(x, y)$ is a "measuring" analytic function, and the integration is taken over the area D of interest. A linear feature vector $\mathbf{l}$ is a vector whose components are linear features (Amari, 1968), i.e.

$$\mathbf{l} = [l_1, l_2, \ldots, l_n]^T \quad \text{where}$$

$$l_i = \iint_D S \, \mu_i \ ,$$

where μ_i is a measuring function for $i = 1, 2, 3, \ldots, n$. The functions μ_i could be any set of "nice" functions, such as moments $(x, y, xy, x^2y,$ etc.), or $e^{i(px+qy)}$, in which case a linear feature corresponds to a Fourier component of the image.

A linear feature is a global image feature, i.e. a global characteristic of the whole image. Using different measuring functions we get different global measurements.

4.2. The constraint

Since there is motion, the induced optic flow field satisfies the following equation (Horn, 1985)

$$S_x u + S_y v + S_t = 0 \tag{12}$$

where (u, v) is the flow at a point (x, y) and S_x, S_y, S_t are the spatiotemporal derivatives of the image intensity function at that point.

Equation (12) can be written as

$$\frac{\partial s}{\partial t} = -\dot{\mathbf{x}} \cdot \nabla S$$

The time derivative of a linear feature vector $\mathbf{l}$ will be

$$\dot{\mathbf{l}} = [\dot{l}_1, \dot{l}_2, \ldots, \dot{l}_n]^T \ ,$$

where

$$\dot{l}_i = \iint \frac{\partial s}{\partial t} \mu_i \, dxdy =$$

$$-\iint \mu_i \, (\dot{\mathbf{x}} \cdot \nabla S) \, dxdy$$

The optic flow field from equation (11) can be written as

$$\dot{\mathbf{x}} = \sum_{k=1}^{n} m_k \, \mathbf{\Phi}_k = \sum_{k=1}^{n} m_k \begin{bmatrix} \Phi_{1k}\,(\mathbf{x}) \\ \Phi_{2k}\,(\mathbf{x}) \end{bmatrix}$$

where $\mathbf{\Phi}_{1k}, \mathbf{\Phi}_{2k}$ are the components of $\mathbf{\Phi}_k$.

On the other hand,

$$\dot{l}_i = -\iint \mu_i \, (\dot{\mathbf{x}} \cdot \nabla S) \, dxdy =$$

$$-\sum_{k=1}^{n} m_k \iint \mu_i \, (\Phi_{1k} \, S_x + \Phi_{2k} \, S_y) \, dxdy$$

$$\text{or } \dot{l} = \sum_{k=1}^{n} m_k \, h_{ik} \quad \text{with}$$

$$h_{ik} = -\iint \mu_i \, (\Phi_{ik} \, S_x + \Phi_{2x} \, S_y) \, dxdy$$

In other words,

$$\dot{\mathbf{l}} = H \cdot m \ , \tag{13}$$

where $H = (h_{ik})$ and $m = (m_1, m_2, \ldots, m_6)^T$ are the motion parameters.

Equation (13) relates the unknown motion parameters to the measurable $\dot{\mathbf{l}}$ and the entries of matrix H which involve the shape parameters, which are assumed to be known.

Of course there is a degeneracy, and we cannot solve for all the motion parameters (only the direction of translation and the rotation).

A simple least squares method can solve for the parameters m_k.

To give an example, let's consider the surface in view to be planar, with equation $Z = pX + qY + c$ with respect to the camera coordinate system. In that case we know p and q. The parameters m_i are $m_1 = \frac{t_1}{c}$, $m_2 = \frac{t_2}{c}$, $m_3 = \frac{t_3}{c}$, $m_4 = \omega_1$, $m_5 = \omega_2$, and $m_6 = \omega_3$, and the functions $\mathbf{\Phi}_k$ are

$$\mathbf{\Phi}_1(\mathbf{x}) = (1 - px - qy, 0)$$

$$\mathbf{\Phi}_2(\mathbf{x}) = (0, l - px - qy)$$

$$\mathbf{\Phi}_3(\mathbf{x}) = \big(-x(1 - px - qy), -y\,(1 - px - qy)\big)$$

$$\mathbf{\Phi}_4(\mathbf{x}) = (xy, y^2 + 1)$$

$$\mathbf{\Phi}_5(\mathbf{x}) = \big(-(x^2 + 1), -xy\big)$$

$$\mathbf{\Phi}_6(\mathbf{x}) = (y, -x) \ ,$$

where the plane is moving with translation $T = (t_1, t_2, t_3)$ and rotation $\Omega = (\omega_1, \omega_2, \omega_3)$.

4.3. Summary

We have presented a method for the correspondenceless recovery of motion, given that the shape is known. The algorithm can be summarized as follows

1. The input is a sequence of images $S\,(x, y, t)$ and $S\,(x, y, t + dt)$.
2. Choose a set of measuring functions $\mu_i\,(x, y)$.
3. Create a linear feature vector $\mathbf{l} = [l_1, l_2, \ldots, l_n]$ where

$$l_i = \iint S(x, y)\, l_i\,(x, y)\, dxdy$$

4. Compute the change of $\mathbf{l}$, i.e. the time derivative $\dot{\mathbf{l}}$, from the images $S\,(x, y, t)$ and $S\,(x, y, t + dt)$.

5 Compute the entries of matrix H. For example, in the planar case

$$h_{13} = -\iint \mu_i \, (\Phi_{13} \, S_x + \Phi_{23} \, S_y) \, dxdy \quad \text{or}$$

$$h_{13} = -\iint \mu_i \, \big(x(px + qy - 1)\, S_x + y\,(px - 1\, qy - 1) S_y\big) \, dxdy$$

6. From equation $\dot{\mathbf{l}} = H\,\mathbf{m}$, solve for $\mathbf{m}$ using a least-squares methodology.

Finally, we wish to stress here the fact that in this algorithm the spatial derivatives of the image intensity function don't need to be computed.[2] This is due to the fact that we use linear features (integration by parts avoids differentiation of the intensity function; instead, the derivative of the measuring function μ_i needs to be computed, and this is well defined). So we don't have to do numerical differentiation, which is an ill-posed problem (Poggio et al., 1986). More importantly, the same approach can be followed when the image intensity function is discontinuous. To further clarify, consider the image to be a dot pattern, i.e. consisting of the points (x_i, y_i), $i = 1, 2, 3, \ldots, n$. Then the moving image is $S(x, y, t) = \sum_{i=1}^{n} \delta\left(x - x_i(t), y - y_i(t)\right)$. In that case a linear feature is trivially computable. So, the algorithm we have presented works for nondifferential image functions as well. We find the concept of linear features (introduced by Amari and communicated to vision researchers by Kanatani) to be very powerful in cybernetics research.

5. THE GENERAL CASE: TOWARDS A THEORY OF OPTIC VOLUME VISION

Here we treat the general case, i.e. the case where an object is moving rigidly and we wish to compute its structure and motion without correspondence, using only one camera. Our input will be the contours in the object (any kind of contour). In other words, the object is considered to consist of curved or straight (non)planar lines. In the case of egomotion such contours can be the zero-crossings in the image, whose computation can be considered robust enough, according to recent literature (Poggio, 1986). Our input will be the time varying image of the contours, i.e. a surface in 3-D space *which is the* Cartesian product of the image (2-D) and time (1-D). In other words, assuming that we can continually accumulate images and we put one after the other, then the image contours will trace a set of surfaces. By examining these surfaces, we can determine constraints among motion, shape and the curvature of the surface. We do not use any intermediate representations such as optic flow or correspondence, and in this way we avoid the aperture problem.

5.1. Mathematical formulation

Keeping our previous nomenclature (3-D coordinates denoted by capital letters and retinal coordinates by small ones), we proceed to mathematically formulate our problem.

A line L in 3-D space can be parameterized in one parameter, i.e.

$$L : \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} X(\sigma) \\ Y(\sigma) \\ Z(\sigma) \end{pmatrix} \quad \text{or} \quad \mathbf{X} = \mathbf{X}(\sigma)\,,$$

under the assumption that this function is differentiable as many times as we wish. But since we are considering a collection of lines, this function has finitely many points of discontinuity, but it does not matter here since we will consider only differentiable points.

The line L is moving rigidly. So, L and $\mathbf{X}$ are functions of time τ, i.e. $L = L(\tau)$ and $\mathbf{X} = \mathbf{X}(\sigma, \tau)$.

Let $\boldsymbol{\omega} = (\omega_1, \omega_2, \omega_3)^T$ be the rotational velocity around the X, Y, and Z axes (see Figure 1) and $\mathbf{T} = (t_1, t_2, t_3)^T$ the translational velocity. Of course the motion is not constant, i.e. $\boldsymbol{\omega}$ and $\mathbf{T}$ are functions of time: $\boldsymbol{\omega} = \boldsymbol{\omega}(\tau)$ and $\mathbf{T} = \mathbf{T}(\tau)$.

The velocity at point $\mathbf{X}$ of the line L is

$$\dot{\mathbf{X}} = \frac{\partial \mathbf{X}}{\partial \tau} = \Omega \mathbf{X} + \mathbf{T}\ , \text{ where}$$

$$\Omega = \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix}$$

The acceleration at point $\mathbf{X}$ is

$$\ddot{\mathbf{X}} = \frac{\partial^2 \mathbf{X}}{\partial \tau^2} = \frac{\partial}{\partial \tau}\frac{\partial \mathbf{X}}{\partial \tau} = \frac{\partial}{\partial \tau}(\Omega \mathbf{X} + \mathbf{T}) =$$

$$= \Omega \dot{\mathbf{X}} + \dot{\Omega} \mathbf{X} + \dot{\mathbf{T}} = \Omega(\Omega \mathbf{X} + \mathbf{T}) + \dot{\Omega} \mathbf{X} + \dot{\mathbf{T}}$$

But, assuming inertial motion (i.e. $\dot{\boldsymbol{\omega}} = \dot{\mathbf{T}} = 0$), we have

$$\ddot{\mathbf{X}} = \Omega \dot{\mathbf{X}}$$

Using perspective projection $\mathbf{P}$, we find that the projection of point

$\mathbf{X} = (X, Y, Z)$ is the point $\mathbf{x} = (x, y)$, with

$$\mathbf{P}(\mathbf{X}) = \mathbf{x} = (x, y) = \left(\frac{X}{Z}, \frac{Y}{Z}\right)\,,$$

where we have assumed the focal length to be unity. The image of a point $\mathbf{X}(\sigma, \tau)$ on the line L is a point $\mathbf{x}(\sigma, \tau)$, i.e. $\mathbf{P}\left(\mathbf{X}(\sigma, \tau)\right) = \mathbf{x}(\sigma, \tau)$. The image of the line $L(\tau)$ will be denoted by $l(\tau)$, a line on the image plane (xy-plane).

So far we haven't *specified how σ should be taken; we* consider it to be the normal coordinate of the line $l(\tau_0)$, i.e. σ is the length from point $\mathbf{x}(0, \tau_0)$ to point $\mathbf{x}(\sigma, \tau_0)$ along the line $l(\tau_0)$.

The shape of a line (say at time τ_0) is the shape of the functions $X(\sigma)$, $Y(\sigma)$ and $Z(\sigma)$, of which the shape of $Z(\sigma)$ is sufficient. (Indeed, since $X = xZ$, $Y = yZ$ with x, y observable from the image, the shape of $Z(\sigma)$ is sufficient.)

So, shape reconstruction means determination of the function $Z(\sigma)$. Our purpose is to obtain $\boldsymbol{\omega}, \mathbf{T}$ and the function $Z(\sigma)$. From now on, we may denote (x, y) as $\mathbf{x}$ or x^i and (σ, τ) as $\boldsymbol{\sigma}$ or σ^i, where the superscripts move from 1 to 2 and x^1, x^2, σ^1 and σ^2 mean x, y, σ and τ respectively.

In the rest of this section we develop some relations that will be of use later. First, let us consider the derivatives of $\mathbf{x}$ at time τ_0. Directly from the definition of σ, $\frac{\partial \mathbf{x}}{\partial \sigma}$ is a unit vector along the line. On the other hand, $\frac{\partial^2 \mathbf{x}}{\partial \sigma^2}$ is a vector perpendicular to the line and its norm represents the curvature of the line Also, $\frac{\partial \mathbf{x}}{\partial \tau}$ is the optic flow

$$\frac{\partial \mathbf{x}}{\partial \tau} = \frac{\partial}{\partial \tau}\mathbf{P}\left(\mathbf{X}(\sigma)\right) = \frac{\partial \mathbf{P}}{\partial \mathbf{X}}\frac{\partial \mathbf{X}}{\partial \tau} = \frac{\partial \mathbf{P}}{\partial \mathbf{X}}(\Omega \mathbf{X} + \mathbf{T})$$

with

$$\frac{\partial \mathbf{P}}{\partial \mathbf{X}} = \begin{bmatrix} \dfrac{\partial P_1}{\partial X} & \dfrac{\partial P_1}{\partial Y} & \dfrac{\partial P_1}{\partial Z} \\[2ex] \dfrac{\partial P_2}{\partial X} & \dfrac{\partial P_2}{\partial Y} & \dfrac{\partial P_2}{\partial Z} \end{bmatrix}$$

[2] However, we need to differentiate in the time dimension.

where $\mathbf{P}(\mathbf{X}) = \left(P_1(\mathbf{X}), P_2(\mathbf{X})\right) = (x, y)$.

Since $\mathbf{P}$ represents perspective projection, we have

$$\frac{\partial \mathbf{P}}{\partial \mathbf{X}} = \begin{bmatrix} \frac{1}{Z} & 0 & -\frac{X}{Z^2} \\ 0 & \frac{1}{Z} & -\frac{Y}{Z^2} \end{bmatrix} = \frac{1}{Z} \begin{bmatrix} 1 & 0 & -x \\ 0 & 1 & -y \end{bmatrix}$$

Also,

$$\frac{\partial \mathbf{x}}{\partial \tau} = \frac{1}{Z} \begin{bmatrix} 1 & 0 & -x \\ 0 & 1 & -y \end{bmatrix} (\Omega \mathbf{X} + \mathbf{T}) = \begin{bmatrix} 1 & 0 & -x \\ 0 & 1 & -y \end{bmatrix} \cdot \left[\frac{\mathbf{T}}{Z} + \Omega \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \right] .$$

On the other hand, we have

$$\frac{\partial^2 \mathbf{x}}{\partial \tau \partial \sigma} = \left[\frac{\partial}{\partial \sigma} \begin{bmatrix} 1 & 0 & -x \\ 0 & 1 & -y \end{bmatrix} \right] \left[\frac{\mathbf{T}}{Z} + \Omega \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \right] +$$

$$+ \begin{bmatrix} 1 & 0 & -x \\ 0 & 1 & -y \end{bmatrix} \frac{\partial}{\partial \sigma} \left[\frac{\mathbf{T}}{Z} + \Omega \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \right] =$$

$$= \begin{bmatrix} 0 & 0 & -\frac{\partial x}{\partial \sigma} \\ 0 & 0 & -\frac{\partial y}{\partial \sigma} \end{bmatrix} \left[\frac{\mathbf{T}}{Z} + \Omega \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \right] +$$

$$+ \begin{bmatrix} 1 & 0 & -x \\ 0 & 1 & -y \end{bmatrix} \left[-\frac{\frac{\partial Z}{\partial \sigma}}{Z^2} \mathbf{T} + \Omega \begin{bmatrix} \frac{\partial x}{\partial \sigma} \\ \frac{\partial y}{\partial \sigma} \\ 0 \end{bmatrix} \right]$$

Furthermore,

$$\frac{\partial^2 \mathbf{x}}{\partial \tau^2} = \frac{\partial}{\partial \tau} \left[\frac{\partial \mathbf{P}}{\partial \mathbf{X}} \dot{\mathbf{X}} \right] = \frac{\partial}{\partial \tau} \left[\frac{\partial \mathbf{P}}{\partial \mathbf{X}} \right] \dot{\mathbf{X}} + \frac{\partial \mathbf{P}}{\partial \mathbf{X}} \ddot{\mathbf{X}} =$$

$$= \frac{\partial}{\partial \tau} \left[\frac{\partial \mathbf{P}}{\partial \mathbf{X}} \right] \dot{\mathbf{X}} + \frac{1}{Z} \begin{bmatrix} 1 & 0 & -x \\ 0 & 1 & -y \end{bmatrix} \Omega \dot{\mathbf{X}}$$

The first term of the above is further simplified to

$$\frac{\partial}{\partial \tau} \left[\frac{\partial \mathbf{P}}{\partial \mathbf{X}} \right] \dot{\mathbf{X}} = \begin{bmatrix} -\frac{\dot{Z}}{Z^2} & 0 & -\frac{\dot{X}}{Z^2} + \frac{2X\dot{Z}}{Z^3} \\ 0 & -\frac{\dot{Z}}{Z^2} & -\frac{\dot{Y}}{Z^2} + \frac{2Y\dot{Z}}{Z^3} \end{bmatrix} \dot{\mathbf{X}} =$$

$$= \frac{1}{Z^2} \begin{bmatrix} -2\dot{X}\dot{Z} + 2X\dot{Z}^2 / Z \\ -2\dot{Y}\dot{Z} + 2Y\dot{Z}^2 / Z \end{bmatrix} = \frac{-2\dot{Z}}{Z^2} \begin{bmatrix} \dot{X} - x\dot{Z} \\ \dot{Y} - y\dot{Z} \end{bmatrix} =$$

$$= \frac{-2\dot{Z}}{Z} \frac{1}{Z} \begin{bmatrix} 1 & 0 & -x \\ 0 & 1 & -y \end{bmatrix} \dot{\mathbf{X}} = \frac{-2\dot{Z}}{Z} \frac{\partial \mathbf{x}}{\partial \tau}$$

But $\frac{\dot{Z}}{Z} = -\omega_2 x + \omega_1 y + \frac{T_3}{Z}$.

In order to improve readability, define matrix A and vector $\mathbf{b}$ as

$$A = \begin{bmatrix} 1 & 0 & -x \\ 0 & 1 & -y \end{bmatrix} \qquad \mathbf{b} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} .$$

These are dependent only on $\mathbf{x}$. Then

$$\frac{\dot{\mathbf{X}}}{Z} = \frac{\mathbf{T}}{Z} + \Omega \mathbf{b}$$

$$\dot{\mathbf{x}} = \frac{\partial \mathbf{x}}{\partial \tau} = A \frac{\dot{\mathbf{X}}}{Z}$$

$$\ddot{\mathbf{x}} = \frac{\partial^2 \mathbf{x}}{\partial \tau^2} = \frac{-2\dot{Z}}{Z} + A\, \Omega \frac{\dot{\mathbf{X}}}{Z} .$$

5.2. $xy\tau$ space: optic volume

This section introduces the idea of the $xy\tau$ space (first originated in Bolles et al., 1987). Suppose that we have taken many images of the object (line) at very short intervals and arranged them as in Figure 3. If we consider the limit where the time intervals go to zero, we will come to the conception of a 3-D space having x, y and τ axes (see Figure 4). In this $xy\tau$ space, a moving (1-D) line is expressed by a 2-D surface (the surface that the line traces as it moves) (Figure 4). It is this surface that we will use as our input. This surface, we assume for the rest of the exposition, is directly observable.

From now on, we will assume that the optic flow $\frac{\partial \mathbf{x}}{\partial \tau}$ is not in the same direction as that of the line at that point; we will only consider such points.

This condition means that the Jacobian is not equal to zero, i.e.

$$\left| \frac{\partial \mathbf{x}}{\partial \boldsymbol{\sigma}} \right| \neq 0 ,$$

that is, there exists an inverse function from $\mathbf{x}$ to $\boldsymbol{\sigma}$ in the neighborhood of $\mathbf{x}$. Note that the parameters $\frac{\partial \tau}{\partial x^i}$, $\frac{\partial^2 \tau}{\partial x^i \partial y^j}$ are observable in the surface, while $\frac{\partial^2 \boldsymbol{\sigma}}{\partial x^i}$, $\frac{\partial^2 \boldsymbol{\sigma}}{\partial x^i \partial x^j}$ are not.

In the $xy\tau$ space, a trajectory of a point is expressed as a line. The above-mentioned surface, i.e. the trajectory of a line, consists of such point-trajectories. However, each point trajectory is not identifiable in the surface; they are mixed up with each other. This is what we call the aperture problem. The mixing of the trajectories leads to an inconvenience (since it is the aperture problem). However, we will show in the next section that the derivatives of the $xy\tau$ surface are directly related to the 3-D motion parameters. So, by just examining the shape of the surface ($xy\tau$) no aperture problem arises, since we treat the $xy\tau$ surface as a "surface" and not as a collection of lines. It is this $xy\tau$ surface that we call the "optic volume", since it contains all the information about 3-D motion and structure.

5.3. The constraint

Clearly, the quantities $\frac{\partial \tau}{\partial x^i}$, $\frac{\partial^2 \tau}{\partial x^i \partial y^j}$ and $\frac{\partial x^i}{\partial \sigma}$ are observable. On the other hand, consider

$$1 = \frac{\partial \tau}{\partial \tau} = \frac{\partial \tau}{\partial x} \frac{\partial x}{\partial \tau} + \frac{\partial \tau}{\partial y} \frac{\partial y}{\partial \tau} \quad \text{or}$$

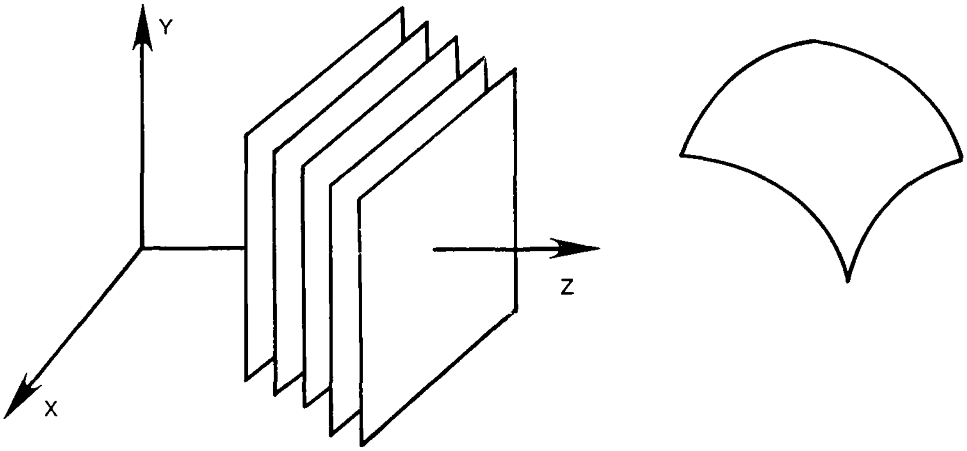


Figure 3.

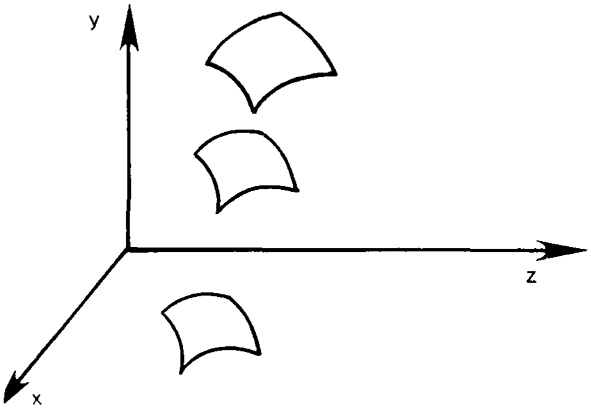


Figure 4.

$$1 = \mathbf{c}^T \dot{\mathbf{x}} \text{ with } \mathbf{c}^T = \left(\frac{\partial \tau}{\partial x} \, \frac{\partial \tau}{\partial y} \right)$$

After differentiating the above relation we obtain

$$0 = \frac{\partial^2 \tau}{\partial \tau^2} = \left[\frac{\partial \tau}{\partial x} \frac{\partial^2 x}{\partial \tau^2} + \frac{\partial \tau}{\partial y} \frac{\partial^2 y}{\partial \tau^2} \right] +$$

$$+ \left[\frac{\partial^2 \tau}{\partial x^2} \left(\frac{\partial x}{\partial \tau} \right)^2 + 2 \frac{\partial^2 \tau}{\partial x \partial y} \frac{\partial x}{\partial \tau} \frac{\partial y}{\partial \tau} + \frac{\partial^2 \tau}{\partial y^2} \left[\frac{\partial y}{\partial \tau} \right]^2 \right]$$

or

$$0 = \mathbf{c}^T \ddot{\mathbf{x}} + \dot{\mathbf{x}}^T D \, \dot{\mathbf{x}} \text{ where } D = \begin{bmatrix} \frac{\partial^2 \tau}{\partial x^2} & \frac{\partial^2 \tau}{\partial x \partial y} \\ \frac{\partial^2 \tau}{\partial y \partial x} & \frac{\partial^2 \tau}{\partial y^2} \end{bmatrix}$$

with $\mathbf{c}^T$ and D observable ($xy\tau$-surface).

So, we have

$$1 = \mathbf{c}^T \dot{\mathbf{x}} = \mathbf{c}^T A \, \frac{\dot{\mathbf{X}}}{Z} = \mathbf{c}^T A \left(\Omega \, \mathbf{b} + \frac{\mathbf{T}}{Z} \right) = \mathbf{\Phi}^T \Omega \, \mathbf{b} + \mathbf{\Phi}^T \frac{\mathbf{T}}{Z} \qquad (5.1)$$

where $\mathbf{\Phi}^T = \mathbf{c}^T A$.
Also,

$$0 = \mathbf{c}^T \ddot{\mathbf{x}} + \dot{\mathbf{x}}^t D \, \dot{\mathbf{x}} = \mathbf{c}^T \left(\frac{-2\dot{Z}}{Z} \dot{\mathbf{x}} + A \, \Omega \, \frac{\dot{\mathbf{X}}}{Z} \right) + \frac{\dot{\mathbf{X}}^T}{Z} A^T DA \, \frac{\dot{\mathbf{X}}}{Z} =$$

$$= \frac{-2\dot{Z}}{Z} + \mathbf{\Phi}^T \Omega \, \frac{\dot{\mathbf{X}}}{Z} + \frac{\dot{\mathbf{X}}^t}{Z} F \, \frac{\dot{\mathbf{X}}}{Z} \quad (\text{where } F = A^T DA)$$

$$= -2 \left(-\omega_2 x + \omega_1 y + \frac{T_3}{Z} \right) + \mathbf{\Phi}^T \Omega \left(\Omega \, \mathbf{b} + \frac{\mathbf{T}}{Z} \right) +$$

$$+ \left(\frac{\mathbf{T}^T}{Z} + \mathbf{b}^T \Omega^T \right) F \left(\Omega \mathbf{b} + \frac{\mathbf{T}}{Z} \right) =$$

$$= \left[-2(-\omega_2 x + \omega_1 y) + \mathbf{\Phi}^T \Omega^2 \mathbf{b} + \mathbf{b}^T \Omega^T F \Omega \mathbf{b} \right] +$$

$$+ \left[-2 \frac{T_3}{Z} + \mathbf{\Phi}^T \Omega \frac{T}{Z} + \frac{T^T}{Z} F \Omega \mathbf{b} + \mathbf{b}^T \Omega^T F \frac{T}{Z} \right]$$

$$+ \frac{T^2}{Z} F \frac{T}{Z} \qquad (5.2)$$

We can write the above equation as

$$f(\Omega) + g(\Omega)^T \, \frac{\mathbf{T}}{Z} + \frac{\mathbf{T}^T}{Z} F \, \frac{\mathbf{T}}{Z} = 0 \qquad (5\ 3)$$

where $f(\Omega)$ is quadratic with respect to Ω and $g(\Omega)^T$ linear. Using (5.1), we eliminate Z from (5.3) and we get

$$(\mathbf{\Phi}^T \mathbf{T})^2 f(\Omega) + (1 - \mathbf{\Phi}^T \Omega \mathbf{b})(\mathbf{\Phi}^T \mathbf{T})^T g(\Omega) \mathbf{T} + (1 - \mathbf{\Phi}^T \Omega \mathbf{b})^2 \mathbf{T}^T F \, \mathbf{T} = 0 \qquad (5.4)$$

Equation (5.4) shows the constraint between the motion parameters and the surface $xy\tau$. Note that scale change in $\mathbf{T}$ is allowed. If we look at n points on the line $l(\tau_0)$, each point i will give values for $\mathbf{\Phi}_i^T$, $f_i(\Omega)$, $\mathbf{b}_i$, $g_i^T(\Omega)$. $\Omega(\tau_0)$ and $\mathbf{T}(\tau_0)$ must satisfy the above constraint for all these points. Thus, the motion parameters at time τ_0 are obtained by solving a system whose i^{th} equation is

$$(\mathbf{\Phi}_i^T\mathbf{T})^2 f_i(\Omega) + (1 - \mathbf{\Phi}_i^T \Omega\, \mathbf{b}_i)(\mathbf{\Phi}_i^T\mathbf{T}) s\ g_i(\Omega)\,\mathbf{T} +$$
$$(1 - \mathbf{\Phi}_i^T \Omega\, \mathbf{b}_i)^2\, \mathbf{T}^T F_i\, \mathbf{T} = 0$$

Clearly, the ability to recover 3-D motion without correspondence (for a motion with zero acceleration) depends on whether or not we can solve the above nonlinear system. This is possible in some cases, but in general, since iterative methods are used, the result of the convergence might be far from the solution. A closed form solution would be very much desirable. On the other hand, a five-dimensional Hough transform (Ballard, 1985) can solve the problem, given that a lot of space and computing power are available.

6. CONCLUSIONS AND FUTURE DIRECTIONS

We have presented a mathematical analysis that indicates that the problem of passive navigation might be solvable without correspondence. This is true in the case where prior information such as depth or shape are known. In the general case, we have outlined a constraint, and the solution comes from solving a nonlinear system. Even though solving such a system is not impossible, it would be desirable to obtain a closed from solution that will enable us to quickly solve for the motion parameters and develop a strict uniqueness analysis of the problem. We consider this problem to be an important future research goal. As to the question "is correspondence necessary for the perception of structure from motion", the answer is "no" in principle, but more research is required in order to establish a closed form solution and a uniqueness analysis. After this is done, the next important question is to discover which approach is more robust: the correspondence-based or the correspondenceless one. Finally, at this point we would like to refer to work by Todd (1985), where it is demonstrated through psychological experiments that the ability of humans to perceive structure from motion is much more general than would be reasonable to expect on the basis of correspondence-based theories, and it is suggested that the computational theories will have to be modified, if they are to account for the high level of generality exhibited by human observers.

REFERENCES

Aloimonos, J. and Brown,C.M., "Direct processing of curvilinear sensor motion under perspective projection," *Proc. Workshop on Computer Vision*, Annapolis, MD, 1984.

Aloimonos, J. and Rigoutsos, I., "Computing 3-D motion without correspondences," *Proc. AAAI*, 1986.

Amari, S., "Invariant structures of signal and feature spaces in pattern recognition problems," *RAAG Memoirs*, **4**, 1968.

Amari, S. and Maruyama, S., "Computation of structure from motion", personal communication.

Ballard, D., "Parameter networks," *Artificial Intelligence*, **22**, 1985

Bolles, R., Baker, H., and Marimont, D., "Epipolar plane image analysis: An approach to determining structure from motion," *IJCV*, **1**, 1987.

Hadamard, J., *Lectures on the Cauchy problem in linear partial differential equations*, New Haven: Yale University Press.

Horn, B.K.P., "Robot Vision," McGraw-Hill, 1985.

Horn, B.K.P. and Schunck, B.G., "Determining optical flow," *Artificial Intelligence*, **17**, 184-204, 1981.

Horn, B.K.P. and Weldon, J., "Robust direct methods for motion recovery", *Proc. 1st International Conference on Computer Vision*, London, UK, 1987.

Huang, T.S., "Direct recovery of motion for planar surfaces", *Digest of Technical Meeting, Optical Society of America*, Lake Tahoe, NV, 1985.

Jenkin, M. and Kolers, P., "Some problems with correspondence", Technical Report 86-10, Dept. of Computer Science, Univ. of Toronto, 1986.

Kanatani, K., "Computation of structure from motion", personal communication, 1985.

Kanatani, K., "Detection of surface orientation and motion from texture by a stereological technique", *Artificial Intelligence*, **23**, 213-237, 1984.

Kanatani, K. and Chou, T., "Recovering 3-D regid motion without correspondence", CAR-TR-297, Computer Vision Laborabory, Center for Automation Research, University of Maryland, College Park, MD, 1987.

Longuet-Higgins, H.C. and Prazdny, K., "The interpretation of a moving retinal image", *Proc. Royal Soc. London*, **B 208**, 385-397, 1984.

Longuet-Higgins, H.D., "A computer algorithm for reconstructing a scene from two projections", *Nature*, **293**, 133-135, 1981.

Negahdaripour, S. and Horn, B.K.P., "Direct passive navigation", *Proc. Image Understanding Workshop*, Miami FL, 1985.

Poggio, T. and Koch, C., "Ill-posed problems in early vision: From computational theory to analog networks", *Proc. Royal Soc. London*, **B**, 1985.

Stewart, G.W., *Introduction to Matrix Computations*, Academic Press, 1973.

Todd, J., "Perception of structure from motion: Is projective correspondence of moving elements a necessary condition?", *J. Experimental Psychology, Human Perception and Performance*, **11**, 689-710, 1985.

Tsai, R.Y. and T.S. Huang, "Uniqueness and estimation of three dimensional motion parameters of rigid objects with curved surfaces", *IEEE Trans. PAMI*, **6**, 13-27, 1984.

Ullman, S., *The Interpretation of Visual Motion*, MIT Press, Cambridge, MA, 1979

Ullman, S., Analysis of visual motion by biological and computer systems," *IEEE Computer*, **14**, 8, 57-59, 1981.

MOTION FROM A SEQUENCE OF IMAGES

Igor Pavlin[1]

Computer & Information Science

University of Massachusetts, Amherst, MA 01003

ABSTRACT

This paper considers the problem of the recovery of motion parameters of a rigid object moving through an environment with constant but arbitrary linear and angular velocities. The method uses temporal information from a sequence of images such as those taken by a mobile robot. Spatial information contained in the images is also used. The temporal sequence, combined with the assumption of constant velocities, provides powerful constraints for the motion trajectory of rigid objects.

We derive a closed form solution for the rigid object trajectory by integrating the differential equations describing the motion of a point on the tracked object. The integrated equations are non-linear only in angular velocity and are linear in all other motion parameters. These equations allow the use of a simple least-square error minimization criterion during an iterative search for the motion parameters. Experimental results demonstrate the power of our method in fast and reliable convergence.

1 INTRODUCTION

1.1 Background

The interest in temporal analysis of image sequences has sharply risen with the increase of available computational power [10]. Bolles and Baker argue that equal weight be given to both temporal and spatial information contained in a sequence of images (frames). Two-image motion algorithms often use spatial integration [18]. However, in a two-image analysis the only temporal information that can be used is differential (positions of image points in two close time instances). This information is too local in scope and too sensitive to errors. In this paper we argue for more extensive use of temporal information. Temporal information is obtained from measurements of distinctive feature positions in the image plane in several time instances (images). A distinctive feature is labeled with both spatial and temporal coordinates and is called an image event. The time coordinate is treated on an equal footing with spatial coordinates. Thus, the idea of smoothing can be applied to time coordinate as well, resulting in reduced sensitivity to small errors in spatial coordinates. Another important feature of our method is the integration of temporal information which is achieved by solving differential equations of motion. By integrating temporal information over several images we get more reliable information about the motion trajectory.

There is evidence that temporal integration is performed by humans, as demonstrated in [21]. It is only the integration of temporal information that is discussed here in greater detail.

Computations of displacement fields [1,6,16,17,22,23,24,38,39] by their nature give more weight to the spatial information in image features. Computations are sensitive to noise in the image because they rely on derivatives of feature displacements. The motion parameters are obtained from a set of high-order non-linear equations, resulting in costly computations, low precision and unstable solutions. The deficiencies of these approaches are especially visible in problems of motion segmentation and occlusion. Motion algorithms using multiple images from an image sequence have been developed by many authors [2,7,10,14,32,34,42].

The research in this paper has been motivated by the problem formulation of Shariat [34] finding the motion parameters of a rigid object moving with constant translational and rotational velocities, under the assumption that five images are equally spaced in time. These assumptions enable Shariat to propose a set of **difference** equations relating the positions of features in the image plane and the motion parameters of the projected point. The solution he obtains is a set of non-linear polynomial equations in unknown motion parameters. The equations are of a rather high (5th) order. The motion parameters are obtained using a Gauss-Newton non-linear least-square method with carefully designed initial-guess schemes.

Our work also relates to that of Broida and Chellappa [13,14] due to the fact that it considers estimation of motion parameters of a rigid body from several images. Broida and Chellappa [14] are estimating kinematic parameters of a rigid body and its structure by tracking several feature points in a sequence of images. The object of their work is to estimate the influence of white noise in feature positions on the recovery of motion parameters and to find the robustness of the method to bad initial guesses. This is important because the point feature extraction methods return feature positions below the expected precision of the algorithm. Their work can be used in the prediction of the feature positions in the next frame(s). It is based on iterative Kalman filtering techniques [12], and stochastic estimation techniques. (good references on the theory behind stochastic estimation techniques used in their work can be found in [20,27]). Wünsche [41] has a similar approach. The prediction of the feature in the next frame usually leads to a reduced search space in the feature correspondence problem, but we have not considered this important issue in the present paper.

[1] Now at the Institute for Robotics and Intelligent Systems at the University of Southern California.

[2] This work was supported by the Defense Advanced Research Agency, DARPA, under grants N00014-82-K0464, and DACA 76-85-C-0008.

Our work concentrates on the computer vision aspect of the problem, and in a different way proves that using several frames is a good way to detect objects moving in the camera's field of view or detect camera movement. We argue, in the sense of Lowe's [25] uniqueness of the viewpoint constraint, that the image positions of the same feature in several frames (i.e., several time instances) provide a powerful constraint on the possible types of object motion. We might call the time constraint "the uniqueness of the *time sequence constraint*". This constraint reflects deterministic motion of a rigid body whose initial position and velocities are known. The time constraint combined with the Lowe's spatial constraint is an excellent tool for detection of rigid body kinematics and shape from motion.

Our approach begins with the set of **differential** equations describing the motion of a point P on a rigid body moving with constant translational and rotational velocity through the environment. The equations are solved analytically, i.e., a set of parametric equations $P = (X(t), Y(t), Z(t))$ describing the helix-like trajectory of the point P is found. The parameters in these equations are the initial position and the linear and angular velocity of the point P. Our goal is to determine these parameters from the central projection of the trajectory of the point P on the image plane.

The central projection of the point P on the image plane is labeled Q. This distinctive feature is characterized by image coordinates x_i, y_i and the time label t_i of the image: $Q_i = (x_i, y_i, t_i)$, $i = 1, \ldots, n$, where n is the number of considered images. We will call Q_i image events. Under the assumptions of the known projective geometry and constancy of motion parameters, only a few image events Q_i are needed to find the motion parameters of the environmental point P. The reason is that we know (from the closed form solution for the trajectory) what constraints exist between image events Q_i and the motion parameters of the point P. The constraints are non-linear only in rotational parameters and the type of non-linearity is known exactly. In our method, there is no constraint imposed on the time-interval between images, and the equations can be more readily adapted for motion of non-rigid objects, as well as for accelerated motions by using perturbation techniques [15,29].

We do not consider the problem of feature correspondence over several images. There are indications that the correspondence problem can be handled quite successfully under the conditions of an approximately known displacement path [3,4,5,8,9]. Bharwani uses correlation measurements and the history of the motion to predict the position of a point feature in the next frame using ths concepts developed by Anandan. This approach has some difficulties in trying to predict the feature position with accuracy less than a half of pixel, due to the shallow nature of the correlation measure at this resolution. The feature prediction was proven to be very helpful both for efficiency and for more reliable feature matching.

Sethi and Jain SET87, for example, use "smoothness of motion" to reduce the search explosion of possible correspondences of several image features through several images. Once a good initial guess for motion parameters is found the search space of correspondences can be drastically reduced and possible correspondences become better defined as the computation proceeds.

We plan to attack the feature correspondence problem by using symbolic features such as lines, and perhaps regions In the case of line matching we can choose feature points to be at the intersection of two lines, and the line correspondence can be established using methods similar to those in Medioni [28]. In this case the major practical problem is to establish the correct line correspondence. The extra effort is rewarded by greater robustness to noise.

The other possibility is to choose lines themselves as features. Spetsakis and Aloimonos [36] use three frames and 13 line correspondences to establish the translational and rotational parameters of the object whose lines are being tracked. They develop a method resulting in a linear system of equations, that authors report to be noise sensitive. We plan to develop a theory that will track the motion of lines through several frames. The theory will have similar structure as the one presented in this paper and it will be a non-linear theory with an explicit frequency dependence. We expect the theory to be less noise sensitive than the one reported by [36].

More recently Williams [40] have developed a spatio-temporal grouping algorithm that produces token-based line correspondences across a sequence of images. This algorithm appears to be robust and utilizes two other algorithms developed at the University of Massachusetts, a spatial line grouping algorithms by Boldt and Weiss [11] and Anandan's algorithm for developing displacement fields with confidence measures [4].

In the case of region matching we can track region centers (like center of mass) as in work by Price [31]. Use of line and region correspondence should result in a more noise-robust approach to the correspondence problem, although we predict some difficulties with the precision of these methods.

One important issue not considered here is the choice of a good initial guessing scheme. In this paper we used simple parameter estimation techniques (such as the sign of the path curvature, direction of motion, magnitude of motion) to predict the initial velocities and position of the tracked feature. We hope to incorporate more sophisticated techniques of initial estimation similar to those seen in work of Shariat [34]. The initial guessing methods used by Lowe [25] in his iterative solution for 2-D to 3-D object recognition are also appropriate for this approach. Lowe's method is also a Newton-Ralphson technique (just like ours) and was proven to be very stable and converged to the right solution in almost all cases. We found that this is the case with our method, too. The good convergence is probably due to the viewpoint and time constraints discussed earlier. Therefore, even with initial guesses that are "far" from the correct solution, the methods converge to the correct solution.

We use a set of noise-free synthetic images to demonstrate the feasibility and speed of the approach, and its high accuracy in noise free conditions. At the time of the publication of this report, we have not obtained results for motion sequences from a natural environment. We expect that the performance will not be as good as reported here, but the use of symbolic features should greatly improve the robustness of the method.

In Section 2 we establish the relevant set of equations, in Section 3 we give a method of solution, in Section 4 we present preliminary results and in Section 5 we discuss further work.

2 Establishing the Equations

In this section, we first derive motion equations for general object motion. We then introduce the assumptions of object rigidity and constant linear and angular velocity and solve the simplified equations in closed form.

Our analysis of general motion [15] is slightly different from the analysis of a camera moving through the environment [24], since the possibility of tracking several objects is kept in consideration. Also, our method of solution is different from that of Shariat [34] and the result is more general. We integrate the differential equations of motion and derive equations that are simpler and more powerful since they contain exact and explicit information about the body motion and they are not restricted to equally-spaced time images.

The camera setup and notation are presented in Figure 1. For the purpose of clarity, we distinguish between 3 coordinate systems: a reference coordinate system with the center at point O (camera-centered coordinate system); an intermediate coordinate system, which is here a center-of-mass coordinate system with the origin C and coordinates (X_c, Y_c, Z_c); and a body-fixed coordinate system (X_b, Y_b, Z_b) with the origin also at the point C. The center-of-mass coordinate system is translating with axes parallel to the axes of the camera-centered coordinate system and the body-fixed coordinate system is rotating around C. The intermediate coordinate system is introduced in order to allow the separation of rotational motion (and perhaps the motion internal to the body-fixed coordinate system) from the rest of the motion. The choice of C is arbitrary for our purposes, and any other point is a valid choice for the center of rotation. $0xy$ is the image coordinate system.

The position of an arbitrary point P on a moving object is characterized according to Figure 1 by vector $R(t)$ (in matrix notation)

$$R(t) = C(t) + r(t) \tag{1}$$

where $C(t)$ is the current position of C and $r(t)$ is the current position of the point P relative to C in both the center-of-mass and body-fixed coordinate systems. Differentation of Eq. (1) gives [15]

$$\left(\frac{dR(t)}{dt}\right)_{camera} = \left(\frac{dC(t)}{dt}\right)_{camera} + \left(\frac{dr(t)}{dt}\right)_{body} + \omega \times r. \tag{2}$$

$\left(\frac{dC(t)}{dt}\right)_{camera}$ is the contribution to the speed of point P coming from the motion of the center-of-mass coordinate system and the term $\omega \times r$ defines changes in the position of the point P due to the instantaneous rotation ω of the body around C. The term $\left(\frac{dr(t)}{dt}\right)_{body}$ is the internal velocity of the point P in the body-fixed coordinate system. This velocity is zero if the body is rigid.

We now introduce the assumptions that the object is rigid, and has constant translational and rotational velocity. The equation of motion of the point P in the camera-centered coordinate system is then the solution of the following pair of equations, describing a constant translational motion V of the origin C and a constant rotational motion ω of the point P around C:

$$\left(\frac{dC(t)}{dt}\right)_{camera} = V \tag{3a}$$

and

$$\left(\frac{dr(t)}{dt}\right)_{camera} = \omega \times \mathbf{r}. \tag{3b}$$

If the translational velocity V is not changing in time then the solution of Eq. (3a) is

$$C(t) = C_0 + V \cdot t. \tag{4}$$

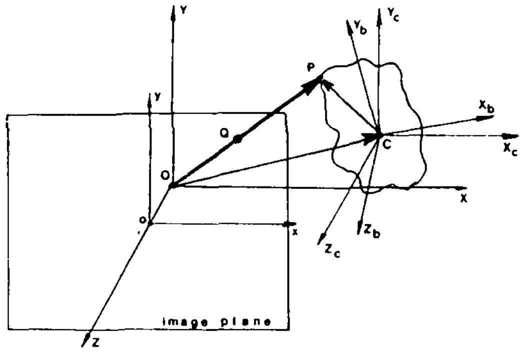


Figure 1: Motion of arbitrary rigid body, and coordinate systems.

C_0 is the initial position of the body-fixed coordinate system.

Let us now derive the solution for the rotational motion, Eq. (3b). For constant angular velocity the solution of Eq. (3b) can be found in several ways. [3] We rewrite Eq. (3b) in a slightly different form

$$\left(\frac{dr(t)}{dt}\right) = \Omega \cdot r \tag{5a}$$

where

$$\Omega = \begin{pmatrix} 0 & -\omega_z & +\omega_y \\ +\omega_z & 0 & -\omega_x \\ -\omega_y & +\omega_x & 0 \end{pmatrix} \tag{5b}$$

is the matrix specifying the rotation of the body. The solution of Eq. (5a) is

$$r(t) = e^{\Omega t} \cdot r_0. \tag{6}$$

r_0 is the initial position of the point P in the body-fixed coordinate system. Here, we make an important observation that the matrix Ω satisfies the relation

$$\Omega^3 = - \| \omega \|^2 \cdot \Omega \tag{7a}$$

where $\| \omega \|$ is the norm of the angular velocity $\omega = (\omega_x, \omega_y, \omega_z)$

$$\| \omega \|^2 = \omega_x^2 + \omega_y^2 + \omega_z^2. \tag{7b}$$

This observation enables us to simplify the matrix equation (6). We find, using Eqs. (7), that the solution of Eq. (5) is

$$r(t) = \left[I + S(\omega, t) \cdot \Omega + C(\omega, t) \cdot \Omega^2\right] \cdot r_0 \tag{8a}$$

where I is the identity matrix and we introduce shorthand notation

$$S(\omega, t) = \frac{sin(\| \omega \| t)}{\| \omega \|}, \tag{8b}$$

$$C(\omega, t) \equiv \frac{(1 - cos(\| \omega \| t))}{\| \omega \|^2}. \tag{8c}$$

Eqs. (8) are similar to Rodrigues formula [19]. The use of functions $S(\omega, t)$ and $C(\omega, t)$ is particularly convenient for small rotations, when these functions are approximately independent of $\| \omega \|$.

The motion of a point P of a rigid body moving with constant translational velocity V and constant rotational velocity ω, with point's starting position R_0

$$R_0 = C_0 + r_0 \tag{9}$$

[3] I thank Mark Snyder for a suggestion that led me to this simple solution.

is then derived by combining the solutions for translational motion (Eq. (4)) and rotational motion (Eq. (8a)):

$$R(t) = R_0 + V \cdot t + \left[S(\omega, t) \cdot \Omega + C(\omega, t) \cdot \Omega^2\right] r_0. \qquad (10)$$

Ω is given by Eq. (5b) and $S(\omega, t)$ and $C(\omega, t)$ are given by Eqs. (8b) and (8c).

Without loss of generality we can assume that $C_0 = 0$, i.e., that the center of rotation and the camera coordinate system coincide at time $t = 0$. [4] With the assumption $r_0 = R_0$ the equation Eq. (10) becomes:

$$R(t) = R_0 + V \cdot t + \left[S(\omega, t) \cdot \Omega + C(\omega, t) \cdot \Omega^2\right] R_0. \qquad (11)$$

This equation describes helix-like trajectory of the point P moving with constant translational and rotational velocity.

There are 9 parameters in Eq. (11) three for each of: the initial position of the point R_0, the translational velocity V, and the rotational velocity ω. The non-linearity in ω is evident from the rotational (third) summand.

Let $Q = (x(t), y(t))$ be the central projection of the point P on the image plane. The components of $R(t) = (X(t), Y(t), Z(t))$ satisfy the following set of equations

$$fX(t) - x(t)Z(t) = 0 \qquad (12a)$$

$$fY(t) - y(t)Z(t) = 0 \qquad (12b)$$

where f is the focal length of the camera, assumed to be the unit of length ($f = 1$). We assume that $Z(t) \neq 0$, for all t. Since there are 9 unknown parameters we need at least 5 images to determine the motion parameters of a single point, P. (Each image supplies two equations for the unknown parameters.) Input parameters are image events $Q_i \equiv (x_i, y_i, t_i) = (x(t_i), y(t_i))$ for some arbitrary times t_i, $i = 0, 1, \ldots, 4$. Some other combination of the number of images and the number of points belonging to the same rigid body can be used as long as there are enough equations to solve for the unknown parameters However, a larger number of images gives a more reliable prediction of motion, and is thus preferred to a large number of points, unless there is a danger of feature disappearance during the time interval considered.

3 Method of Solution

In this section we use a generalized version of Newton's iterative method to solve the equations for the motion parameters which we developed in the previous section . We can foresee some of the advantages of the proposed solution, Eq. (11). The known type of non-linearity makes the method converge fast and be more stable. It is quite possible that there is an especially suitable numerical procedure for this type of non-linearity, although we have not found one yet.

The initial position of the point P is determined by the 3 parameters X_0, Y_0, and Z_0. Since Eqs. (12) are homogeneous equations of the first order in $X(t), Y(t)$, and $Z(t)$, the solution is determined up to a scale factor - usually referred to as the loss of depth during the central projection. We can set the scale factor equal to an arbitrary constant, Z_0, supplied in practice by some other method (e.g., from laser range data). The first image in the sequence then provides enough information to determine the two unknown parameters

$$X_0 = x_0 Z_0, \qquad (13a)$$

$$Y_0 = y_0 Z_0. \qquad (13b)$$

Thus, we are left with six unknown parameters labeled as a group

$$\xi \equiv (V_x, V_y, V_z;\ \omega_x, \omega_y, \omega_z) \qquad (14)$$

for which we need at least three more images. In the work presented here we were not concerned with the determination of the structure of the moving body. One method for determining the structure can be found in the work by Broida and Chellappa [14]. Their method results in eight unknown parameters and computes the structure of the moving body.

The closed form of the solution enables us to simplify the formulation of the minimization problem in that the error becomes a linear function of point coordinates. Namely, assuming no over-determined system of equations, we can use a generalized Newton iterative procedure [37] in the form

$$J(\xi_n) = (\xi_{n+1} - \xi_n) \cdot \epsilon(\xi_n), \qquad n = 0, 1, \ldots \qquad (15)$$

where the error of the solution is

$$\epsilon(\xi) = \begin{pmatrix} X(t_1) - x_1 Z(t_1) \\ X(t_2) - x_2 Z(t_2) \\ X(t_3) - x_3 Z(t_3) \\ Y(t_1) - y_1 Z(t_1) \\ Y(t_2) - y_2 Z(t_2) \\ Y(t_3) - y_3 Z(t_3) \end{pmatrix} \qquad (16)$$

ξ_{n+1} is a new, better set of values for the unknown motion parameters. $J(\xi_n)$ is the Jacobian matrix

$$J_{ij}(\xi_n) = \frac{\partial \epsilon_i}{\partial \xi_n^j}, \qquad i, j = 1, \ldots, 6 \qquad (17)$$

which is easily computable from Eqs. (16). Note that since we already know the explicit dependence of the coordinates of P on the motion parameters, we do not have to minimize expressions of the form

$$\epsilon_i^1 = \frac{fX(t_i)}{Z(t_i)} - x_i \qquad (18)$$

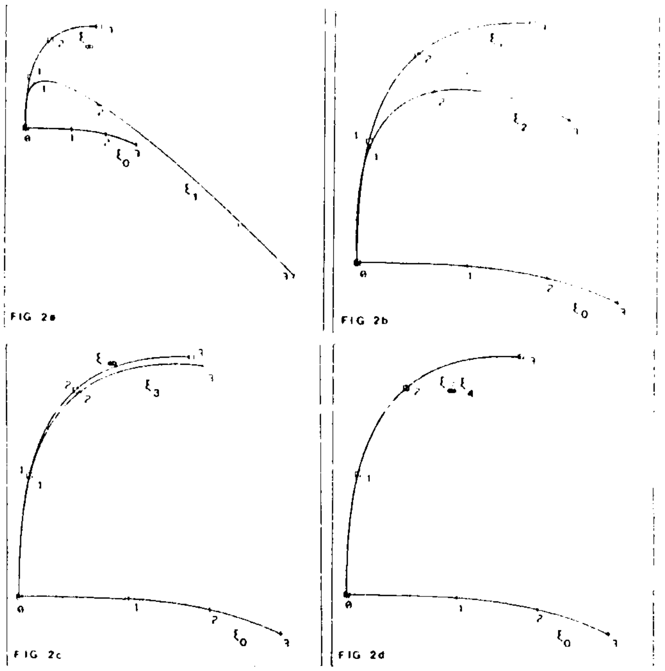
often found in optic flow (differential) computations. By formulating the minimization problem in this way, Eqs. (16)), we avoid additional non-linearity of equations and are headed for faster and more stable convergence.

4 Experimental Results

In this section we give preliminary results for experimental runs designed to test the ideas and feasibility of our approach. The results demonstrate the generality of the method and its very fast convergence.

The algorithm is implemented in Common Lisp on a TI-Explorer. The algorithm takes only a few seconds to compute motion parameters. The sequence of images in Figure 2 shows centrally-projected trajectories after each step in Newton's iterative procedure for a point moving with constant linear and angular velocity. In each of the images in Figure 2 there are three different types of trajectories. One, labeled with squares, represents the correct (goal) trajectory. Each square is centered around the image coordinates (x_i, y_i), of the point moving with

[4] The search for parameters is slightly more difficult when this assumption is not used.



Four iterative steps in the recovery of motion parameters for the trajectory labeled by squares. Crosses label initially guessed trajectory ξ_0, triangles label trajectories after the step s (trajectories ξ_s, $s = 1,2,3,4$) of the Newton iteration method described in text . Squares label the correct solution ξ_∞.

Figure 2: Convergence of Iterative Method in Recovery of Motion Parameters.

the correct set of motion parameters $\xi = (V;\ \omega)$. Image events $Q_i = (x_i, y_i, t_i)$, $i = 0,1,2,3$, are provided as input parameters to the iterative algorithm. An initial guess for motion parameters ξ_0 produces the trajectory labeled with crosses. The trajectory which represents the motion with improved motion parameters ξ_s, $s = 1,2,3,\ldots$ (after each step s of iteration) is labeled by triangles. Note that the initial position of the point R_0 is found, according to Eqs. (13), from the first pair of image coordinates (x_0, y_0). This is the reason that all the trajectories start from the same initial point (labeled as "0").

To demonstrate the generality of the method, we choose to detect motion of an arbitrary linear and angular velocity. Figure 2 illustrates experiments which detect motion of a point on a body moving with linear velocity $V = (0.1, 0.2, 0.3)$ and angular velocity $\omega = (0.3, -0.2, 0.2)$, i.e.,

$$\xi_\infty = (0.1, 0.2, 0.3;\ 0.3, -0.2, 0.2).$$

The unit of length is $f = 1$ and the unit of time is 1.

Convergence to the correct solution is highly dependent on the choice of the initial guess. Even a very rough and easily obtainable estimate of the motion parameters (e.g., the sign of the z-component of the linear and angular velocity) drastically reduces the amount of search. As can be seen in Figure 2, our initial guess $\xi_0 = (0.0, 0.3, 0.4;\ 0.2, 0.0, 0.2)$, is "far" from the correct solution

$$\|\xi_0 - \xi_\infty\| / \|\xi_\infty\| = 50.8\%,$$

but has the same sign of the curvature and direction of motion in one direction. After the first iteration we get

$$\textit{step 1}: \quad \xi_1 = (0.21, 0.42, 0.48;\ 0.50, -0.36, 0.25). \quad (\textit{Figure 2a})$$

This is shown as the trajectory labeled with triangles in Figure 2a. In each of the subsequent figures we show the current trajectory (triangles) after the second, third, and fourth iteration, together with the initially guessed, and the correct, trajectory.

$$\textit{step 2}: \quad \xi_2 = (0.07, 0.25, 0.47;\ 0.33, -0.25, 0.25), \quad (\textit{Figure 2b})$$

$$\textit{step 3}: \quad \xi_3 = (0.09, 0.21, 0.30;\ 0.31, -0.20, 0.21), \quad (\textit{Figure 2c})$$

$$\textit{step 4}: \quad \xi_4 = (0.099, 0.201, 0.300;\ 0.300, -0.200, 0.201). \quad (\textit{Figure 2d})$$

Thus, after only 4 iterations the relative error in the motion parameters is

$$\|\xi_4 - \xi_\infty\| / \|\xi_\infty\| = 0.3\%.$$

Note, in particular, that components V_x and ω_y were first initialized to 0 but are still correctly computed. Good convergence was also found for many other types of motions and initial guesses.

5 Discussion

As expected for any set of non-linear equations, there are initial guesses for which the convergence is poor and the expected solution is not found. The situation can be compared with finding the roots of the equation $z^3 - 1 = 0$ in the complex plane. Depending on the initial guess any of the three roots can be found. Although the solution is usually the root which is the closest to the initial guess, there are some cases where a different root is found.

The problem can be sometimes resolved using over-constrained systems (more image events than necessary). We are currently working on exploring the relation between the type of motion and the value of the initial guess, in order to devise appropriate initial guessing schemes.

In a forthcoming paper we will present a detailed analysis of the algorithm for both synthetic and real images, and for several types of motion. We are presently exploring a number of important issues, before applying the method to a sequence of real images. One problem is the feature correspondence over several images. A larger distance between image features will better define the trajectory and recover motion parameters more accurately. Unfortunately, distant features are more difficult to correlate between images. A strength of our method is that it allows arbitrary time intervals between successive images. Allowing larger time intervals between images facilitates the separation of different motions with similar projections.

Another issue is the sensitivity of the solution to the positional error of the image features. The idea of smoothing [30] applied to the time coordinate, together with error analysis similar to [14,26,35] could be used as a promising start. The Newton iterative method is easily generalized to over-determined systems, so as to insure greater robustness to noise. A related problem is the non-uniqueness of solutions of the non-linear equations. This is the case, for example, when the trajectory has a much larger $\| \omega \|$ than expected, but passes through the same space-time events, a phenomenon that could be called the "stroboscopic" effect. [5]. In a sense it is an undersampling in the time domain, that could be corrected. Over-determined systems have a potential to solve this problem as well. If we can assume that angular velocity is small, then the solution space of the iteration procedure can be restricted and multiple solutions can be avoided.

If the rotational frequencies are very high, a completely different problem arises: it becomes increasingly hard to track the motion of a single feature due to its periodic disappearance. Our approach is suitable for high frequencies since it does not require the constant time interval between the time frames. However, some estimation techniques of the feature positions will have to be incorporated.

We also plan to investigate the motion of several objects simultaneously, as well as that of several features on a single object. Tracking of several features on a single object can produce an estimate of the 3-D structure of the object or it can be used to facilitate motion segmentation. Another very interesting direction for investigation is the derivation of equations (in a manner similar to the one presented in this paper) for structures more complex than a point, i.e., to lines, planes, and other surfaces of a rigid body [25,36,40]. Finally, using perturbation techniques these equations can be generalized to motions of bodies with slowly changing shape, and/or slowly changing motion parameters.

5.1 Conclusion

In conclusion we have used a spatio-temporal analysis of image events to present a quite general, robust and computationally efficient method for the recovery of motion parameters of moving objects under the formulation posed by Shariat [34] of constant motion. The closed form solution (containing exact and explicit information about the body motion) has other advantages: solution strategies can be adapted to the known type of non-linearity resulting in faster and more reliable convergence; the initial guessing scheme can exploit constraints derived from the proposed solution; a generalization to more images, more features, and/or variable time intervals between images is readily available; and, finally, the constraint of constant motion parameters can be relaxed, using perturbation techniques for slowly varying parameters.

ACKNOWLEDGMENTS

I would like to thank Edward Riseman for encouragement and careful readings of this article. Thanks also go to Jasmina Pavlin for careful readings of this paper and useful suggestions, and to Allen Hanson, Mark Snyder, P. Anandan and Deborah Strahman for useful comments.

6 References

REFERENCES

[1] G. Adiv, *Determining 3-D Motion and Structure from Optical Flow Generated by Several Objects.* Proceedings of Image Understanding Workshop, DARPA, October 1984, pp. 113-129.

[2] J. K. Aggarwal, and A. Mitiche, *Structure and Motion from Images.* Proceedings of Image Understanding Workshop, DARPA, Miami Beach, Florida, December 1985, pp. 89-97.

[3] P. Anandan, *A Computational Framework and an Algorithm for the Measurement of Visual Motion.* To appear in the International Journal on Computer Vision, 1988.

[4] P. Anandan, *Measuring Visual Motion From Image Sequences.* Ph.D. Dissertation and Technical Report 87-21, Department of Computer and Information Science, University of Massachusetts, Amherst, March 1987.

[5] P. Anandan, *A Unified Perspective on Computational Techniques for the Measurement of Visual Motion.* Proceedings of the International Conference on Computer Vision, London, England, June 1987.

[6] P. Anandan, *Computing Optical Flow From Two Frames of an Image Sequence.* Technical Report 86-16, Department of Computer and Information Science, University of Massachusetts, Amherst, Massachusetts 01003, April 1986.

[7] D. H. Ballard, and O. A. Kimball, *Rigid Body Motion from Depth and Optical Flow.* Computer Vision, Graphics, and Image Processing, **22**, 1983, pp. 95-115.

[8] S. Bharwani, E. Riseman, and A. Hanson, *Refinement of Environment Depth Maps Over Multiple Frames.* Proceedings of Image Understanding Workshop, DARPA, December 1985, pp. 413-420.

[5] remark by Yigal Gur

[9] S. Bharwani, E. Riseman, and A. Hanson, *Refinement of Environmental Depth Maps Over Multiple Frames.* Proceeding of the IEEE Workshop on Motion: Representation and Analysis, Charleston, SC, May 1986.

[10] R. C. Bolles and H. H. Baker, *Epipolar-Plane Image Analysis: A Technique for Analyzing Motion Sequences.* IEEE Proceedings of the 3rd Workshop on Computer Vision: Representation and Control, Bellaire, Michigan, October 1985, pp. 168-178.

[11] M. Boldt and R. Weiss, *Token-Based Extraction of Straight Lines.* To appear in the International Journal on Computer Vision, 1988.

[12] S. M. Bozic, *Digital and Kalman Filtering.* Edward Arnold, London, 1979.

[13] T. J. Broida and R. Chellappa, *Estimation of Object Motion Parameters from Noisy Images.* Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, San Francisko, California, June 1985, pp. 90-99.

[14] T. J. Broida and R. Chellappa, *Kinematics of a Rigid Object from a Sequence of Noisy Images: A Batch Approach.* Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, 1986, pp. 176-182.

[15] A. L. Fetter and J. D. Walecka, *Theoretical Mechanics of Particles and Continua.* McGraw-Hill, Inc., New York, 1980.

[16] J. J. Gibson, *The Perception of the Visual World.* Riverside, Cambridge, Mass, 1950.

[17] E. C. Hildreth, *Computation of the Velocity Field.* Proc. R. Soc. Lond., 1984, **B 221**, pp. 189-220.

[18] B. K. P. Horn and B. G. Schunck, *Determinining Optical Flow.* MIT, A.I. Memo 572, April 1980. Also, Artificial Intelligence, 1981, **17**, pp. 185-203.

[19] B. K. P. Horn, *Robot Vision.* The MIT Press, Cambridge, Massachusetts, 1986, p. 450.

[20] A. H. Jazwinski, *Stochastic Processes and Filtering Theory.* Academic Press, New York, 1970.

[21] G. Johansson, *Spatio-Temporal Differentiation and Integration in Visual Motion Perception.* Psych. Res., **38**, 1976, pp. 379-383.

[22] J. J. Koenderink and A. J. van Doorn, *Local Stfucture of Movement Parallax of the Plane.* J. Opt. Soc. Am. **66**, No. 7, July 1976, pp 717-723.

[23] D. T. Lawton, *Processing Dynamic Image Sequences from a Moving Sensor.* Ph. D. Dissertation, University of Massachusetts, Amherst, MA 01003. Also, Technical Report 84-05, Department of Computer and Information Science, University of Massachusetts, Amherst, Massachusetts 01003, February 1984.

[24] H. C. Longuet-Higgins and K. Prazdny, *The Interpretation of a Moving Retinal Image.* Proc R Soc Lond., 1980, **B 208**, pp 385-397

[25] D. G. Lowe, *Three-Dimensional Object Recognition form Single Two-Dimensional Images.* Technical Report No. 202, Courant Institute, New York, February 1986.

[26] D. H. Marimont, *Inferring Spatial Structure from Feature Correspondences.* Ph.D. Dissertation, Stanford University, March 1986.

[27] P. S. Maybeck, *Stochastic Models, Estimation and Control.* Vols. 1-3. Academic Press, New York, 1982.

[28] G. Medioni and R. Nevatia, *Matching Images Using Linear Features.* IEEE Transactions on Pattern Analysis and Machine Intelligence, **PAMI-6**, November 1984, pp. 675-685.

[29] P. M. Morse and H. Feshbach, *Methods of Theoretical Physics, II.* McGraw-Hill, New York, 1953.

[30] I. Pavlin, E. Riseman, and A. Hanson, *Translational Motion Algorithm with Global Feature Constraints.* Technical Report 86-58, Department of Computer and Information Science, University of Massachusetts, Amherst, Massachusetts 01003, December 1986.

[31] K. Price, *Relaxation Matching Techniques - A Comparison.* IEEE Transactions on Pattern Analysis and Machine Intelligence, **PAMI-7**, 1985, pp. 617-623.

[32] J. W. Roach and J. K. Aggarval, *Determining the Movement of Objects from a Sequence of Images.* IEEE Transactions on Pattern Analysis and Machine Intelligence, **PAMI-2**, November 1980, pp. 554-562.

[33] I. K. Sethi, and R. Jain, *Finding Trajectories of Feature Points in a Monocular Image Sequence.* **PAMI-9**, January 1987, pp. 56-73.

[34] H. Shariat, *The Motion Problem: How to use more than Two Frames.* Ph. D. Dissertation. University of Southern California, Los Angeles. Also, Technical Report IRIS 202, Institute for Robotics and Intelligent Systems, University of Southern California, Los Angeles, CA 90089-0273, October 1986.

[35] M. Snyder, *The Accuracy of 3D Parameters in Correspondence-Based Techniques.* Technical Report 86-28, Department of Computer and Information Science, University of Massachusetts, Amherst, Ma 01003, July 1986.

[36] M. E. Spetsakis and J. Aloimonos, *Closed Form Solution to the Structure From Motion Problem from Line Correspondences.* Proc. AAAI-87, Seatle, WA, July 1987, pp. 738-743.

[37] F. Stummel and K. Hainer, *Introduction to Numerical Analysis.* Scottish Academic Press, Edinburgh, 1980.

[38] R. Y. Tsai and T. S. Huang, *Estimating Three-Dimensional Motion Parameters of a Rigid Planar Patch.* Proc. PRIP, 1981, pp. 94-97.

[39] A. M. Waxman and S. Ullman, *Surface Structure and 3-D Motion from Image Flow: A Kinematic Analysis.* Technical report CS-TR-1332, Carnegie-Mellon, October 1983

[40] L. Williams and A. Hanson, *Translating Optical Flow Into Token Matches*. To appear in DARPA Image Understanding Workshop, Cambridge, MA, April 1988.

[41] H. J. Wünshe, *Detection and Control of Mobile Robot Motion by Real-Time Computer Vision*. SPIE Conference 727 on "Mobile Robots", Cambridge, MA, October 1986.

[42] B. L. Yen and T. S. Huang, *Determining 3-D Motion and Structure of a Rigid Body Using the Spherical Projection*. Computer Vision, Graphics, and Image Processing, **21**, 1983, pp. 21-32.

Recognizing Animal Motion

Nigel H. Goddard [1]
Computer Science Dept., University of Rochester
and
Hughes Artificial Intelligence Center

Abstract

We address the problem of recognition of animal motion from pure motion cues. A graphical representation for animal gait is introduced, and its connectionist implementation given. Assuming principal views, a method of indexing into this gait representation from low level invariants is described. Preliminary results from the first implementation are presented. The approach is motivated and constrained by biological considerations.

I. Introduction

The human visual system has a remarkable ability to discriminate between different types of movement. The classic illustration of this ability is Johannson's Moving Light Display (MLD) [Johansson, 1973, Johansson, 1976]. Reflective pads were placed at the joints of an actor dressed in black, and the actor illuminated. Films were taken of the actor walking, jumping and making various other movements against a black backdrop. When these films were shown to subjects, they all recognized the display to be of a person walking, jumping, *etc.*, but reported single frames to be meaningless patterns of dots. The time required for this act of recognition was on the order of a perceptual event: a presentation time of no more than 200 msec was sufficient for all subjects to make the correct discrimination. Further experiments [Kozlowski and Cutting, 1977, Cutting and Kozlowski, 1977] demonstrated the sensitivity of this faculty: subjects could determine the actor's gender, and could even identify the actor if (s)he was known to the subject.

This paper reports the early results of an attempt to produce a computational account of this capability, consistent with the psychological and neurophysiological literature. A crucial aspect of the model is the representation and processing of temporal factors. We assume an input representation that could reasonably be expected to be available from lower levels in the visual pathway. The representation of memory and the mechanism for indexing into it is expressed as a connectionist network. The output is a pattern of activation over a small part of the memory network, indicating which object/movement has been recognized.

II. Motion Processing in Primate Visual Systems

Visual motion is in essence unlike other visual information. Color, stereopsis, texture and brightness are all static qualities, which we can experience immediately. Visual motion is a temporal quality. Although our experience of it seems immediate, it has sometimes been treated as a perception constructed from memory of a sequence of images. By now it is clear that motion is in fact a fundamental biological sense (for overviews, see [Nakayama, 1985] and [Maunsell and Newsome, 1987]).

There are two obvious ways the motion information available in the Johannson experiments could be used to generate the percepts of *person* and *walking*, or indeed the single percept of *walking person*. The first method would be to use the motion information to index directly into memory, implying a memory representation rich in temporal information. This method places motion information in a central position *vis-a-vis* the recognition process. The second method would use the motion information to reconstruct various static qualities of the scene object (such as structure), and use those static qualities to index into memory and recognize the object. Having recognized the object, the motion of various key parts of the object could be used to discriminate between gaits. In this second method, the motion information is used in two ways: to recover static qualities; and to disambiguate a small number of gaits.

In this paper we address the first method, but do not rule out the second. The underlying view is that the visual system can use multiple sources of information which are processed in parallel by separate but interacting systems. No one source or system is *required* for recognition; each one alone may be *sufficient*. Each recognition process indexes into a memory structure tailored to the type of information it uses, the different memory structures for the same object being linked. Although pure motion information may also be used to derive structural information which in turn may drive the form recognition process, we argue here for a recognition process operating directly on

[1] The author's current address is Hughes Artificial Intelligence Center, 23901 Calabasas Road, Calabasas CA 91302.

the motion information and indexing into a memory structure devoted to the representation of motion.

Such a motion-specific process and memory structure must play a role in MLD experiments. MLD's presented upside down are not recognized, whereas moving stick figures presented upside down are recognized [Feldman, 1988a]. If the hypothesized structure-from-motion process is independent of memory, it should work equally well with upside down MLD's, deriving upside down stick figures that are recognizable. Since these upside down MLD's are *not* recognizable, either there is a separate motion recognition process, or the structure-from-motion process is dependent on top-down feedback, which would imply a memory structure rich in motion information. Brain damaged patients provide further evidence for separate processes. Lesions to the temporal lobe can lead to the inability to identify faces, while leaving intact the ability to identify from body motion [Damasio *et al.*, 1982]; and lesions to parietal cortex can impair recognition from body motion while leaving object recognition unimpaired.

III. Computational Issues

As with any biological recognition problem, the fundamental computational issues are: what are the tractably computable *invariants* in the input?; what is the memory *representation* for ecologically significant items in terms of these invariants?; and how are the input invariants used to *index* into the memory representation?

We have already established that the representation of animal gait in memory will be complete in itself; it will not presume any particular representation of the static object. It is not possible however to describe the movement of a non-rigid object without at least implicitly describing the structure of the object. We will refer to a sequence of movements of co-ordinated parts of an animal as a *scenario*, using Feldman's term in a restricted fashion [Feldman, 1988b].

Animal motion is not arbitrary. Across a wide range of animals it can be characterized by pendular motion: the upper arm rotates about the shoulder, the lower arm about the elbow, the hand about the wrist, *etc.* We may abstract the skeletally based motion as that of a jointed stick figure. Moreover animal motion is generally planar. When walking or running, most limb movements are in the plane whose perpendicular is defined by the direction of gravity and the direction of overall motion. Dancing is an example of a human movement that is truly three dimensional.

A. Restricting the Problem Space

We make the following assumptions. The motion to be recognized is that of an articulated stick figure with bright spots at the joints, moving parallel to the image plane and viewed orthogonally. If the trunk of the figure is moving, we assume the imaging system is tracking the center of rotation of the trunk, so that in the image the trunk is undergoing pure rotation. The limbs are rotating about an end of the trunk, and so on. Thus we have a movement which can be completely described by the length of each stick in the figure, and the change in angle at each joint over time. We shall assume that the change in joint angle over time is piecewise linear, i.e. a sequence of segments of constant angular velocity.

We have now delimited the class of movements in such a way that a complete representation is possible. Starting with the trunk, we assign a level to each part (stick) of the figure. The level is given by the number of joints between the part and the trunk. Thus a thigh would be at level one and a calf at level two. Then the motion of the figure is described by the motion of each joint between a part at a lower level and a part at the next highest level. Each of these joints undergoes a sequence of constant angular velocity changes, which for biological movements such as walking is periodic. The fundamental motion *event* under these assumptions is a change in angular velocity. The set of sequences *together with information co-ordinating them* describe the motion completely: sufficiently to unambiguously regenerate it. Such a set of sequences of events constitute a *scenario*.

A sample abstract scenario is given in Figure 1. In this diagram the column on the right (b) shows an abstract stick figure initially in the shape of the numeral four, undergoing periodic motion. Over the course of ten time steps, as noted by the column on the left, it articulates a cyclical motion, the horizontal stick (the "trunk") remaining stationary and the other three sticks rotating about its ends. The middle three columns each show one of these three sticks rotating about the horizontal stick. The solid line drawings indicate an event occurring, the dotted line drawings are merely for illustration. In this scenario, there are three parts each of which undergoes four events in the ten step period. At time step 10, the object is back in its original shape, ready to repeat the cycle once again.

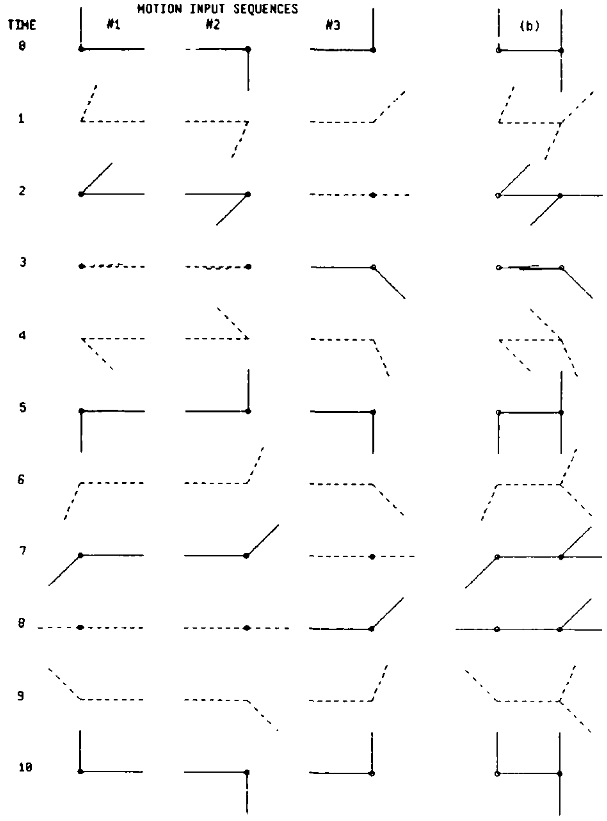


Figure 1: *folding-four* scenario

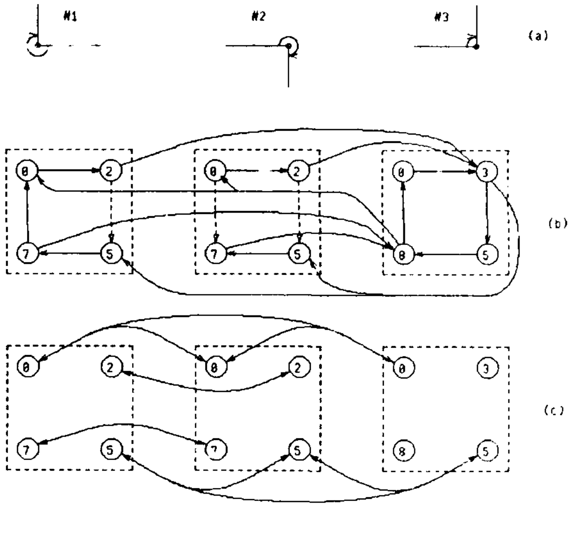


Figure 2: **scenario graph**

B. Low-level visual invariants

What kind of input data can we expect to be available for indexing into the gait memory? The stimulus is a moving pattern of small bright spots against a dark background. We assume a mid-level process computing the Visual Vector Analysis [Johansson, 1973], giving as output an encoding of a moving stick figure. The vector analysis process continuously computes, for each joint, the angle and angular velocity at that joint. For example the angle and angular velocity of the lower leg relative to the upper leg would be continuously available. As in the related Tinker Toy recognition project [Cooper and Hollbach, 1987], we assume a principal views treatment and features which are invariant to scale. As viewpoint changes, so that motion is no longer parallel to the image plane, these angular position and velocity cues vary little and can be considered invariant for the purposes of indexing.

Why should we assume these particular cues for indexing? We have ample evidence of cells tuned for orientation [Allman *et al.*, 1985, Rodman and Albright, 1988]. Combining the output of two or more of these would make the angle at the joint available. There are cells sensitive to rotation [Saito *et al.*, 1986, Sakata *et al.*, 1985], although not sufficiently highly tuned for particular velocities. However the output of several broadly tuned cells can be combined to achieve finer tuning. The strongest evidence for motion cues based on velocity is that people can estimate position and velocity well for a variety of tasks, but further derivatives (acceleration, *etc.*) poorly.

C. Representing Scenarios

A simple graphical representation follows from the specification of a scenario given above. We represent each event, or point (in time) when the angular velocity changes, by a graph node. The nodes are labeled with the new angular velocity and the absolute angle of the joint at that time. Directed edges between nodes represent sequence, each edge being labeled with the time between the two nodes. Each sequence is represented by such a graph. The graph is cyclical if the sequence is periodic. The graphs for the sequences are linked with directed edges that specify the *co-ordination* between the sequences, using labels on the edges as before.

Figure 2 shows the scenario graph for our *folding-four* example described previously in Figure 1. Row (a) shows the three moving parts of the object and the direction in which angle and angular velocity are measured. Rows (b) and (c) show the co-ordination within and between the three sequences. For clarity, each event node is labeled by the time step at which it occurs. The dotted boxes are for illustrative purposes only. The event nodes in the leftmost boxes are associated with joint #1, those in the middle boxes with joint #2, and so on. Row (b) shows the unidirectional edges indicating sequentiality, and row

(c) shows the bi-directional edges indicating simultaneity. Each event node shown in (b) is repeated for clarity in (c). The edges are not labeled with their associated time delay; it is the difference between source and destination node times.

We have assigned graph nodes to those points in time when something significant is happening, namely a change in angular velocity for one of the joints. This covers the significant points for pure motion information (given our assumptions above). But even with simple stick figures there are moments during the motion which are significant cues, but significant for reasons other than motion. The simplest example for stick figures is co-linearity - when the joint angle is 0 or 180 degrees. For richer images including, for example, color and shading, we may expect many more of these non-motion significant points. These are also events and can be included as new nodes in the graphical representation. The difference is that these new events are not labeled with angular velocity but with color, shading, *etc.* In this paper we will ignore such non-motion events.

A problem with this representation is that it does not allow for time-scaling. However if we re-interpret the edge labeling so that the time differences between nodes are not absolute but assumed to be a multiple of some time interval, and we have some way of determining the base time interval, then a single graph will represent a scenario executed at any speed. This is an avenue to be explored in the future.

The graphical representation developed above is naturally implemented as a connectionist network[2]. Each graph node is represented by a unit, and each directed labeled edge by a link with an associated time-delay. The units have a site for *priming* activation which arrives along these delay links, and another site for input from the lower levels of the visual system. Initially all units receive a small amount of priming activation. Units expect activation to arrive at both sites simultaneously. If priming activation or input activation arrives, but not both, then the events in the image are not corresponding to the scenario represented. If the image events do indeed correspond to the scenario represented, then priming activation should flow through the network, building up as it does so. For periodic motions this activation should saturate quickly. Figure 2 is easily re-interpretable as a connectionist network.

A scenario is recognized when activation flows around the network representing it. It is a simple matter to attach a network to the scenario network to detect when and how strongly activation is flowing through the scenario network. The output of this *evaluation* network is a measure of how similar the input is to this particular scenario. The details of this evaluation network are not given here, but would be important in any quantitative study.

D. Indexing

Assuming the input described above, i.e. at each time step, for each joint in the image, a readout of the angle and angular velocity at that joint, how do we index into scenario memory? We would like the time-complexity of the indexing algorithm to be independent of the number of scenarios stored in memory. However the time-complexity may be dependent on the complexity of the scenarios, where complexity is loosely described by the number of body parts in motion and the number of events the parts transit. We would like the indexing algorithm to be tolerant of missing data points (for example, due to occlusion), and to incrementally converge on the correct scenario as more and more data arrives. At the same time we must avoid exponential growth in the number of units and links required as the number of scenario memories increases. We would also like to be able to take advantage of evidence based on structural or other static qualities of the object if it is available.

Using units to detect *changes* in angular velocity, our input may be easily discretized into a set of sequences of events at which angular velocities change for each joint. These sequences are exactly analogous to the sequences of events represented by the nodes in the scenario graph. For a given scenario we must match the input sequences against the stored sequences to determine which input sequence corresponds to which stored joint sequence. Not only must a mapping from input to scenario be established, the *co-ordination* between input sequences must match the co-ordination between joint sequences in the scenario. We must perform this match for each scenario memory. If we assume a solution to the first problem (matching a particular scenario against the input), then we can achieve recognition time independent of the number of scenarios stored in memory at a cost of linear increase in the number of units and links: we match against all scenario memories in parallel. This is trivial to do in a connectionist network; we simply duplicate the matching machinery for each scenario.

E. The Correspondence Problem

Solving the correspondence problem is harder. We cannot wait until we have all the data before attempting to match. This must also be an incremental process over time. Our approach is to attempt to match all input sequences against all stored sequences in parallel. Again it is trivial to achieve parallel matching in a connectionist network, if one is willing to pay the price in terms of the number of units and links required. If n is the maximum number of joint sequences for any one gait, then we will require *n-squared* binding networks for each gait in order to match everything in parallel. Since typically the stick figures have only a few joints (less than 10), this is not too high a price to pay.

Figure 3 gives a schematic outline of the architecture we adopt to solve the correspondence problem. This dia-

[2] In our networks units have one or more *sites* at which links arrive, and where input activation is processed. This enables differential treatment of inputs.

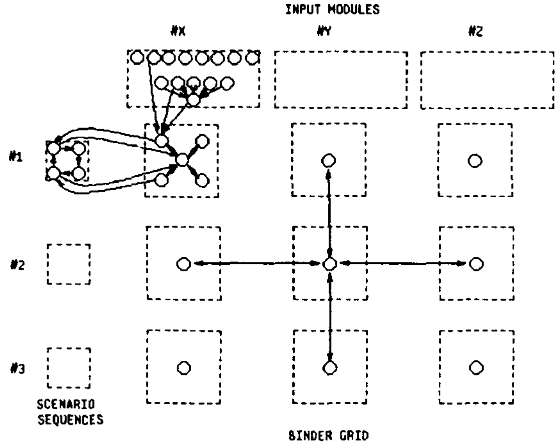


Figure 3: **indexing architecture**

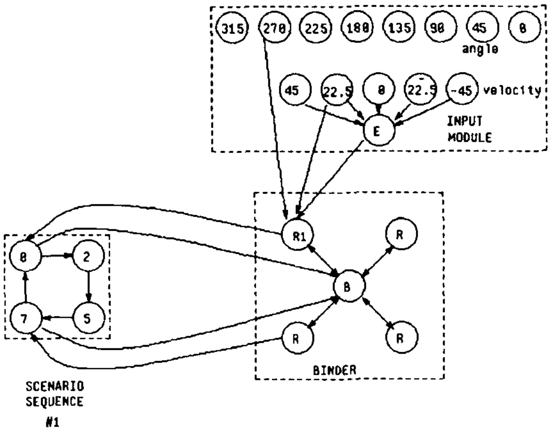


Figure 4: **binding network details**

gram shows our example abstract scenario in memory in the left hand column (the dotted boxes correspond to those in Figure 2). The input modules are shown in the top row. The *n* by *n* grid of binding networks for matching is shown in the middle. For each scenario there is a separate scenario memory network and another set of binding networks. The input modules are not duplicated, the single set providing input for all binding networks.

The input is a time-varying pattern of activation over the set of input modules. There is one input module for each joint in the scene (see Figure 4 for details). Each input module consists of a set of *angle* and a set of *angular velocity* units. The angle is that between the major and minor parts at the joint, and the angular velocity its derivative with respect to time. Thus the time-varying pattern at an input module should match the expected pattern represented in one of the scenario sequences (#1, #2 or #3 in the *folding-four* example). The scenario closest to the motion in the scene is found by matching each input module against each scenario sequence. Not only must each active input module match a sequence, all active input modules must match *different* sequences in the same scenario, and the co-ordination between events in the different input modules must match the co-ordination between sequences in the scenario.

Keeping in mind that our correspondence problem is to match input modules against scenario sequences, to perform the comparison in parallel we use the grid of binding networks, shown in Figure 3. The input modules pass events to the binding networks, which pass them on with varying degrees of activation to the scenario sequences. At the same time the binding networks are comparing the activation from the input modules with the activation in the scenario sequences. Any particular binding network is connected to one input module and one scenario sequence. If the activation from both correspond (i.e. arrive at the same time), then the *strength* of binding increases; otherwise it decreases. If the input scene corresponds to the scenario, then over time a consistent set of binding networks will become active, all others becoming inactive; and the scenario itself will become very active. If the match is partial, parts of one or more sets of consistent binding networks will become active to some degree; and the scenario will become active to some degree.

Figure 4 shows the details for one binding network. At the top is an input module. It has a set of units tuned to a particular angular range and another set tuned to particular angular velocity. Unit *E* is an event detecting unit, connected to all the velocity tuned units in the module; it fires when the angular velocity changes, our definition of an event. At the left of Figure 4 is a sequence network, sequence #1 from Figure 2. In the middle is a binding network.

How do these binding networks work? A binding network performs two functions: it passes on input events from its input module to its sequence network; and it compares the events occurring in the input module with those occurring in the sequence network. Events are differentiated by the angle and the angular velocity at the joint, and occur when the angular velocity changes. For each event represented in the sequence network, there is a *relay unit*, labeled *R* in Figure 4, in the binding network. This units fires when the corresponding event occurs in the input module (detected by unit *E*). For example, unit *R1* in Figure 4 is tuned to fire when the joint angle is 270 degrees (angle unit input), and the angular velocity changes (unit *E* input) to 22.5 degrees per time step (velocity unit input). It sends activation modulated by the *binding level* to the appropriate event unit in the sequence network. The *binding level* is the activation of the *binding* unit in the network, labeled *B* in Figure 4. It has a site for each event represented in the sequence network. A site receives activation from a network event unit and from the corresponding relay unit. If a site receives input from the relay unit, it expects input soon after from the sequence network. Otherwise there is a mis-match occurring. The binding unit checks that the sites are receiving activation in this fashion, and if so increases its activation level. Otherwise its activation decreases.

All binding networks connected to the same input module have their binding units arranged in a mutually inhibitory network (see Figure 3). Similarly all binding networks connected to the same sequence network are arranged so that their binding units inhibit each other. Thus, even with locally ambiguous input, so long as globally the input is determinate, the correct scenario should be the most highly activated. If evidence for matching is available from other sources, for instance form or color matching, it can be used to influence the scenario match by providing input to the binding units in the binding networks.

F. Preliminary Results

The architecture has been implemented using the Rochester Connectionist Simulator [Goddard *et al.*, 1988] for the single scenario described above. The results depend on the actual parameters used in the activation functions and in the time-delayed links. As expected, presenting perfect input causes the scenario network to saturate quickly (within one cycle of the motion). Presenting imperfect input, e.g. with one of the input modules inactivated to simulate occlusion, caused the scenario network to activate more slowly. Overall it is clear that the architecture solves the problem, and moreover that it can be tuned along several dimensions: speed, sensitivity to missing data, sensitivity to incorrect data. Exactly how the network should be tuned is a matter for further research, and will require psychophysical experiments.

IV. Related Work

Much work has been done on static object recognition ([Lowe, 1985]), and on the use of motion information to reconstruct object information [Ullman, 1979] that could be used for recognition. Yet surprisingly little attention has been paid to the use of motion information directly in recognition.

A. MLD Computation

[Rashid, 1980] shows how the trajectory information used as input for the model presented here can be computed from real MLD image sequences, but does not seriously address the problems of matching this information against a database of models, the focus of this paper. [Hoffman and Flinchbaugh, 1982] shows that with the assumption of planar motion, two "structure from planar motion" propositions may be proved, but again the indexing problem is not addressed.

B. Sequence Recognition with Networks

There has been little work on temporal sequences in connectionist network research. [Chun, 1986] outlines a representation for temporal sequence in massively parallel networks. His focus is the representation of temporal constraints, using single units to represent events and *constraint* units to mediate the links. The model has been successfully used for word recognition, and is similar to the one developed in this paper.

[Kleinfeld, 1987] and [Sompolinksy and Kanter, 1986] have developed sequence storage and retrieval models using spin-glass type networks. This type of network, after initialization to some state, evolves through stable attractor states at each iteration, the states forming the sequence. The temporal qualities are achieved through links with associated delays. These models are extensions of the Hopfield type [Hopfield, 1982], and rely on two kinds of connections: symmetric links to encode attractor states, and slow asymmetric links to encode transitions between states. Sequence recognition in [Kleinfeld, 1987] is achieved by adding input units with connections to the memory units. Activation from these input units is required to complete transitions from one state to the next.

V. Conclusions

We have introduced a representation for articulated stick figure motion that is naturally implemented in a massively parallel network. A network architecture for indexing into this memory representation from biologically plausible input has been designed. The results of the preliminary implementation and tests are encouraging.

Our immediate task is to generate more abstract scenarios and test the ability of the architecture to discriminate between scenarios. Further ahead it will be necessary to design a way of integrating indexing into scenario memory with indexing into static form memory. We are considering conducting informal perceptual psychology experiments to obtain more insight into human processing capabilities.

VI. Acknowledgements

Inspiration for this work came from Jerome Feldman at the Department of Computer Science, University of Rochester. Funding was generously provided by Hughes Aircraft Company. My colleagues at Hughes Artificial Intelligence Center critiqued a previous draft of this paper. Christof Koch at the California Institute of Technology, Division of Biology, gave financial support and help with the biological literature.

References

[Allman *et al.*, 1985] John Allman, Francis Miezin, and EveLynn McGuinness. Direction- and velocity-specific responses from beyond the classical receptive field in the middle temporal visual area (mt). *Perception*, 14:105–126, 1985.

[Chun, 1986] Hon Wai Chun. A representation for temporal sequence and duration in massively parallel networks: exploiting link interactions. In *Proceedings of AAAI-86*, August 1986.

[Cooper and Hollbach, 1987] Paul R. Cooper and Susan C. Hollbach. Parallel recognition of objects comprised of pure structure. In *Proceedings DARPA Image Understanding Workshop*, pages 381–391, February 1987.

[Cutting and Kozlowski, 1977] J.E. Cutting and L.T. Kozlowski. Recognizing friends by their walk: gait perception without familiarity cues. *Bull. Psychonometric Soc.*, 9(5):353–356, 1977.

[Damasio *et al.*, 1982] A. R. Damasio, H. Damasio, and G. W. Van Hoesen. Prosopagnosia: anatomical basis and behavioral mechanisms. *Neurology*, 32:331–341, 1982.

[Feldman, 1988a] Jerome A. Feldman. Personal communication, January 1988.

[Feldman, 1988b] Jerome A. Feldman. *Time, Space and Form in Vision.* Technical Report, Department of Computer Science, University of Rochester, 1988.

[Goddard *et al.*, 1988] Nigel H. Goddard, Mark A. Fanty, and Kenton Lynne. *The Rochester Connectionist Simulator.* Technical Report TR213, Department of Computer Science, University of Rochester, 1988.

[Hoffman and Flinchbaugh, 1982] D.D. Hoffman and B.E. Flinchbaugh. The interpretation of biological motion. *Biological Cybernetics*, 42:195–204, 1982.

[Hopfield, 1982] John J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings National Academy of Science*, 79:2554, 1982.

[Johansson, 1973] Gunnar Johansson. Visual perception of biological motion and a model for its analysis. *Perception and Psychophysics*, 14:201–211, 1973.

[Johansson, 1976] Gunnar Johansson. Spatio-temporal differentiationn and integration in visual motion perception. *Psychological Research*, 38:379–393, 1976.

[Kleinfeld, 1987] David Kleinfeld. Sequential state generatiohn by model neural networks. *PNAS*, 83, 1987.

[Kozlowski and Cutting, 1977] L.T. Kozlowski and J.E. Cutting. Recognizing the sex of walker from dynamic point-light displays. *Perception and Psychophysics*, 21(6):575–580, 1977.

[Lowe, 1985] David G. Lowe. *Perceptual Organization and Visual Recognition.* Kluwer Academic Publishers, Boston, Mass., 1985.

[Maunsell and Newsome, 1987] John H. R. Maunsell and William T. Newsome. Visual processing in monkey extrastriate cortex. *Annual Review of Neuroscience*, 10:363–401, 1987.

[Nakayama, 1985] K. Nakayama. Biological image motion processing: a review. *Vision Research*, 25(5):625–660, 1985.

[Rashid, 1980] Richard F. Rashid. *Lights: A system for the interpretation of moving light displays.* PhD thesis, University of Rochester, 1980.

[Rodman and Albright, 1988] Hillary R. Rodman and Thomas D. Albright. Coding of visual stimulus velocity in area mt of the macaque. *Vision Research*, in press, 1988.

[Saito *et al.*, 1986] Hiue-aki Saito, Masao Yukie, Keiji Tanaka, Kazuo Hikosaka, Yoshiro Fukada, and Eiichi Iwai. Integration of direction signals of image motion in the superior temporal sulcus of the macaque monkey. *Journal of Neuroscience*, 6(1):145–157, 1986.

[Sakata *et al.*, 1985] H. Sakata, H. Shibutani, K. Kawano, and T. L. Harrington. Neural mechanisms of space vision in the parietal cortex of the monkey. *Vision Research*, 25:453–463, 1985.

[Sompolinksy and Kanter, 1986] H. Sompolinksy and I. Kanter. Temporal association in asymmetric neural networks. *Physical Review Letters*, 57(22):2861–2864, December 1986.

[Ullman, 1979] S. Ullman. The interpretation of structure from motion. *Proceedings of the Royal Society*, 203:405–426, 1979.

Issues in Extracting Motion Parameters and Depth from Approximate Translational Motion

R. Dutta R. Manmatha Edward M. Riseman M. A. Snyder

Computer and Information Sciences Department
University of Massachusetts at Amherst

Abstract

In dynamic imaging situations where the sensor is undergoing primarily translational motion with a relatively small rotational component, it might seem likely that *approximate* translational motion algorithms can be effective in determining depth. By restricting the processing to the two dimensions of translational motion there is a great reduction in complexity from the five dimensions of general motion (excluding the scaling component of sensor velocity or displacement). In an attempt to recover depth of points over a sequence of frames in such situations a set of three algorithms was applied sequentially: elimination of the effect of small rotations, then extraction of the FOE, and then determination of depth of points. The goal was the extraction of obstacles at a distance beyond the accuracy of the range sensors in the ALV research program.

In this paper we show, however, that even small rotations can significantly affect the performance of algorithms designed for pure translational motion. In addition, a theoretical analysis of the problems in computing the FOE is presented. The attempt to correct for small rotations in the motion system described above was ineffective. Accurate extraction of the FOE and recovery of depth without dealing with general motion was not achieved.

More recently, a pair of previously developed algorithms have been combined to yield a *general motion algorithm* and have been applied to sequences of approximate translational motion. By determining the small rotational components of motion, more promising depth results have been extracted under the same circumstances described above. The conclusion is that in order to determine depth of points in many real situations, general motion analysis must be applied even when sensor motion is primarily translational. Various alternatives for extracting depth from motion are briefly considered.

1 Introduction

1.1 Background - Motion Algorithms Developed at the University of Massachusetts

At the University of Massachusetts several algorithms have been developed and applied to real scenes in an attempt to develop practical and robust techniques for mobile vehicle applications. The goals have been the recovery of sensor motion parameters when they are unknown, the determination of vehicle location via landmarks in known environments, and the recovery of the depth of environmental points, both for the derivation of mechanisms for obstacle avoidance, and for object recognition.

In cases of pure translational motion of the sensor, Lawton [1,2] developed reasonably robust techniques for recovery of the sensor motion parameters (or equivalently the focus of expansion - FOE); these techniques for FOE search were extended by Pavlin [3,4]. On several synthetic image sequences, the FOE was recovered within 1-2 degrees (about 5–10 pixels) for axes inside a 45 degree cone around the line of sight and within 5-10 degrees outside that range [3]. It has also been successfully applied to several natural scenes.

In another investigation the goal was to recover depth from *known* motion. Since the exact motion parameters could never be known, some degree of imprecision should be included. Snyder [5] has provided an analysis of the effects of uncertainty in the location of the FOE and the tracked feature points on the determination of depth, and on the location and size of the search window in future frames, as a function of the uncertainties in the FOE, the feature points, and the corresponding environmental depths. Bharwani [6] applied these ideas using the time-adjacency relationship in an iterative spatio-temporal strategy of prediction-refinement of feature point displacements and their associated depths. The algorithm was applied to a sequence of hand-registered real-world images, with either an FOE that was hand-supplied or extracted from the Lawton-Pavlin algorithm.

In several cases this algorithm achieved useful depth values over a sequence of images [6,7]. Thus, the methods of FOE determination and depth extraction, used together, showed strong promise for practical vehicle navigation in outdoor environments.

In addition to the above a two-stage algorithm for analyzing general motion has also been implemented. The first stage is based on Anandan's hierarchical extraction of displacement vectors [8,9,10]. The second stage is based on Adiv's algorithm [11,12] for analyzing general motion in the presence of multiple independently moving objects.

1.2 The Problem - Obstacle Avoidance via Motion of an Approximately Translating Mobile Vehicle

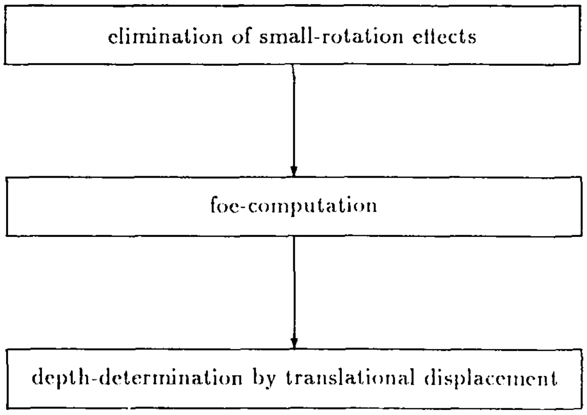
The algorithms evaluated in this paper are part of the research effort directed towards the development of a system for obstacle avoidance via motion analysis as part of the Autonomous Land Vehicle (ALV) project. The primary goal of the ALV project is to design a vehicle which can autonomously navigate through a wide range of environmental situations. For obstacle avoidance the depth of objects out to about 80 feet needs to be extracted to determine whether they may be obstacles. The phase-difference based ERIM scanning laser range finder [13] seems to be limited to distances of less than 40 feet Thus our specific research goal was to extract depth at distances greater than the effective range of the ERIM sensor, or approximately in the 40-80 foot range.

In an effort to provide more powerful constraints that would enhance the effectiveness of such a practical application, we decided to consider the case of depth from *known* translational motion. Since precisely known sensor motion is not possible without sophisticated instrumentation, the problem was modified to one of *approximate* translational motion.

However, in actual application of the methods to the Carnegie-Mellon University NAVLAB [13] moving down a path that is approximately horizontal and planar, certain general problems occur. The vehicle may bounce, sway, vibrate, and of course turn slightly while still primarily undergoing translation, and the road while approximately planar may vary with minor undulations, pits, and bumps. The net effect when examining sequences of images taken at 2 or 4 foot intervals makes clear that there is significant "misregistration" between frames beyond the image displacements expected from smooth translational sensor motion. The expected translational displacement (i.e. the translation due to the *average* or *smoothed* motion) between any pair of frames can be significantly affected by perturbations which involve both a translational and rotational component. This could result in unpredictable movement of the "FOE"[1] over the sequence of frames (i.e. there might be a varying direction of translational motion which if "averaged" across frames would be the "approximate" translational motion), as well as a varying rotational component that causes additional rotational displacements of image points independently of the depths of the associated environmental points. Thus, even though the vehicle is translating in some fixed direction that remains approximately constant across many frames, there can be significant local variations between nearby pairs of frames in the sequence. In order to obtain accurate depths from translational motion, it may be necessary to update the FOE location and remove the effect of rotational displacements.

1.3 An Attempt at Building a Translational Motion Subsystem

Figure 1: Processing of Approximate Translational Motion for Depth Extraction


elimination of small-rotation effects
foe-computation
depth-determination by translational displacement

As a consequence of the effects described in the previous section, the two translational motion algorithms for FOE recovery [1,3] and extraction of depth [6] mentioned earlier cannot be applied directly. Either some form of preprocessing is necessary to allow translational processing to be maintained despite the problems described, or else the complexities of general motion must be considered. An attempt was made to carry out "registration" of the images [14] i.e., removal of small rotations by translation of the image as a simple preprocessing step to allow

[1] If there is rotation, there is of course no FOE; nevertheless, the translation vector will intersect the image plane somewhere and sometimes we will use the term FOE in this manner.

effective translational motion analysis. The three algorithms shown in Figure 1 were to be applied sequentially to determine the depth of environmental points. The first algorithm was intended to eliminate a significant portion of small rotations by image registration [14]. The second algorithm was to compute the position of the FOE for the registered images [2,4]. The third algorithm was to use the time-adjacency relationship and an iterative refinement procedure to compute depth [6].

1.4 Analysis of the Problems

In view of all the difficulties with the estimation of motion parameters, the difficulty of autonomous navigation through unknown terrain is not surprising. Our restriction of the problem to finding the depth of points from approximately known translational motion was intended to introduce sufficient constraints to overcome these problems. It is the purpose of this paper to draw attention to the practical difficulty of developing such a system. While each of the algorithms in isolation had some experimental success in achieving its intended goal, they failed as an integrated set of algorithms. A global system is much more difficult to construct because of cumulative errors generated by the subsystems. In this paper we show that even small amounts of rotation can cause large errors in determining the FOE and depth using translational motion algorithms. Simple rotation-elimination mechanisms, such as the one used here, often place too many restrictions on the imaging situations for practical use.

The effect of errors on the estimation of depth and motion parameters has been studied by many authors [5,15,16,17,18]. Tsai and Huang [18] demonstrated experimentally that "unless the error in finding point correspondences is less than 3%" the solutions to the motion parameters are "overwhelmed by noise." They also assert on the basis of experimental evidence that "error of the estimated motion and geometrical parameters can be reduced by using more point correspondences only if the error for the latter is less than 3%." Fang and Huang [17] showed that as the distance between the object and the image plane increases it becomes more difficult to solve the motion equations proposed by them. It must be noted that these studies are by no means general and essentially apply to the algorithms proposed or studied by the researcher. However, they provide us with insight regarding the difficulty of motion parameter and depth estimation.

As a consequence of the difficulty of recovering the depth points in practical situations of approximate translational motion, we return later in this paper to an experimental investigation of the case of general motion. We consider a two stage algorithm for analyzing general motion: Anandan's hierarchical extraction of displacement vectors [8,9,10] followed by Adiv's algorithm [11,12] for analysis of general motion in the presence of independently moving objects. Initial results show the combination of these two algorithms to provide reasonably good estimates for the depth of environmental points.

2 Depth From Approximate Translational Motion

Recovery of depth from approximately known translational motion has been studied by Bharwani, et al [6]. This scheme is based on the time-adjacency relation and involves a prediction-based iterative refinement scheme for computing increasingly more accurate depth maps. Since it is based on the time–adjacency relation (see equation (1)), the position of the FOE must be known before the algorithm can be applied. In the next section, we give a brief discussion of the various components of the approach as developed at the University of Massachusetts.

2.1 FOE determination

The basic algorithm for determining the FOE was developed by Lawton [2]. The position of the FOE is determined in essentially two phases. In the first phase distinctive features are extracted under the assumption that they can be more easily tracked across frames. In the second phase the direction of translation is found in the following manner. Initially a position of the FOE is hypothesized in the image plane. Selection of an FOE constrains each feature point to appear in the second frame on the radial line emanating from the hypothesized FOE and passing through the corresponding feature point in the first frame. The point on the path that correlates best (up to some resolution of correlation matching with the feature point in the first frame) will be assumed to be the feature correspondence, and the deviation from perfect correlation is regarded as an error. The sum of these errors for the set of feature points gives the total error for the hypothesized FOE. The objective is to find the FOE position for which the total error is minimized. A search algorithm involving a coarse sampling of the FOE followed by local hill climbing was demonstrated to be effective on several natural scenes.

In Pavlin, et al.'s modification of Lawton's algorithm, the orientation of the axis of translation and associated error values are represented in polar coordinates on the unit sphere centered at the camera focus. This gives a more uniform sampling of the hypothesized FOE positions than working with the image plane. The error surface is constructed by fitting a smooth surface over the values produced by a coarse sampling over polar angles. The sampling is then repeated locally around the minimum of the coarsely sampled error surface. This hierarchical search along with smoothing was intended to make it robust and more efficient. The smoothing was intended to eliminate fluctuations in the error surface and reduce the need for finer sampling resolution, thereby reducing the required computation.

Experimental results have shown that this algorithm is effective on real image data if

1. the motion is purely translational, and
2. approximately 8 to 16 distinctive feature points can be found which can be reasonably tracked across frames.

However, for reliable results when no a priori information is available to place constraints on the processing (e.g. approximate location of the FOE) the algorithm is quite time consuming. In addition various design parameters must be adjusted to account for the worst eventualities (e.g., the initial FOE sampling cannot be made too coarse because it might result in the selection of an incorrect global minimum for the FOE). Nevertheless, this algorithm appears to be quite useful for images in many practical situations.

2.2 Depth Determination

The discussion here describes the algorithm proposed and implemented by Bharwani, et al [6]. Consider the *time-adjacency* [19] *relation:*

$$\frac{D(t)}{d(t)} = \frac{Z(t)}{W(t)}, \qquad (1)$$

where

d(t) : the radial velocity(displacement) in pixels of a point P in the image

D(t) : the distance of the point P from the FOE in pixels

W(t) : the Z-component of the actual velocity(displacement) of the corresponding scene point.

Z(t) : the depth of the corresponding scene point from the camera. Hence, depth Z can be found via pure translational motion if we know W, the FOE position, and the matches across frames.

Essentially the technique can be described as iteratively improving the precision of depth estimates over a *sequence* of frames by using the depth estimates (along with their associated uncertainties) from the previous time steps. In future frames, the match region can be predicted better, if the following are known:

(a) the current estimate of the depth of a point

(b) the uncertainty associated with the depth.

Therefore, the method matches points between frames at a particular match resolution, finds depth and uses this to predict a smaller match window to be searched with a finer match resolution. Increasing the match resolution was intended to give a more accurate depth. In addition, the method can be guaranteed to have a constant upper bound on computation for processing between frames by controlling the correlation resolution (via interpolation) of the feature point matchings as an inverse function of the size of the predicted search window. This makes it a simple scheme for depth determination with a sequence of images for translational egomotion.

Even when the motion of the camera is known, ambiguities in the matching process between frames leads to erroneous depth determination. The matching process might fail because of occlusion, highlights, shadows, distortion of the surface over time, and similarity of image features etc. In addition to these problems, an insufficient search area for the match along the displacement path can result in no matches or incorrect matches being found. The matching problem is a very widely studied area in motion analysis [9,20,21] and will not be discussed here.

The major problems with the depth determination algorithm are:

1. The correspondence problem (i.e. effectiveness of the match process);
2. The accuracy of FOE determination;
3. The effect of small rotations;

A discussion of the problems of FOE determination and effect of small rotations is given in the next two sections and is a key focus of this paper.

3 Problems of FOE determination

3.1 The Image Model

We present here the equations describing displacement fields which will be used throughout the paper. Let

- (X, Y, Z) be a Cartesian coordinate system with its origin at the nodal point of the camera,
- (x, y) represent the corresponding coordinate system of a planar image,
- f be the focal length of the camera,
- F_i be the planar image of the environment at time t_i,
- F_{i+1} be the planar image of the environment at time t_{i+1}.

An environmental point $P(X, Y, Z)$ then appears at the point $p(x, y)$ in the image F_i. If x and y are given in pixels, then

$$x = f\frac{X}{Z},$$

$$y = f\frac{Y}{Z},$$

where f is the focal length of the camera in pixels, given by

$$f = \frac{N}{2}\cot\frac{FOV}{2} \tag{2}$$

Here $FOV$is the field of view of the camera, and the image resolution is $N \times N$.

Let the motion relative to the camera of a point in the rigid environment have translational component $\vec{T} = (U, V, W)$ and rotational component $\vec{\Omega} = (A, B, C)$. It can be shown that if

1. W/Z is much smaller than 1 (i.e., the translation of the point in the Z direction is much smaller than the depth of the point);
2. the rotation parameters are small; and
3. the field of view is of the camera is not very large;

then the displacement vector (u,v) of the image point between frames F_i and F_{i+1} is given by

$$\begin{aligned} u &= \frac{fU - xW}{Z} + (\frac{-Axy + Bx^2}{f}) + Bf - Cy, & (3)\\ v &= \frac{fV - yW}{Z} + (\frac{-Ay^2 + Bxy}{f}) - Af + Cx. & (4) \end{aligned}$$

We note that u and v both have translational and rotational components:

$$(u, v) = (u_t, v_t) + (u_r, v_r) \tag{5}$$

where

$$u_t = \frac{fU - xW}{Z} \tag{6}$$

$$v_t = \frac{fV - yW}{Z} \tag{7}$$

$$u_r = (\frac{-Axy + Bx^2}{f}) + Bf - Cy \tag{8}$$

$$v_r = (\frac{-Ay^2 + Bxy}{f}) - Af + Cx \tag{9}$$

3.2 Overview

The *motion field* is the projection of the 3-D velocity/displacement field onto the image plane. Given the motion field, the motion parameters (U, V, W, A, B, C) and the scene structure (Z) can be uniquely determined up to a scale factor - except for a two-way ambiguity in certain special cases [22].

The *optical flow* is the image velocity or displacement recovered from the image. If the optical flow is precisely equal to the *motion* field, the *motion* parameters and scene structure recovered are the correct ones. In general, the optical flow is not equal to the motion field. This may be due to *noise*, *measurement errors* (e.g. usually our displacements are only accurate to within 0.5 pixel), correspondence error, etc. Even small errors in the optical flow can produce motion parameters which are far from the correct *ones* [11].

In the following subsections we first establish the error in the FOE position as a function of the error in the image displacements. By interpreting small rotations as deviations/errors from the displacements derived from pure translation, we show that even small rotations may cause large errors in the FOE position returned by pure translational methods. As a corollary we show how the different feature points should be weighted to improve the FOE computation.

3.3 The Effect of Displacement Errors on FOE Position

Let us assume that the motion is purely translational. Then from equation (5) it follows that

$$\begin{aligned} u = u_t &= \frac{fU - xW}{Z} = f\frac{U}{Z} - x\frac{W}{Z}, & (10)\\ v = v_t &= \frac{fV - yW}{Z} = f\frac{V}{Z} - y\frac{W}{Z}. & (11) \end{aligned}$$

It is well known that the flow field given by (10) and (11) is purely radial, with center at the FOE. The position of the FOE (when it exists) is the point where $u = v = 0$. From (10,11) we easily see that this is the point

$$(x_0, y_0) = (f\frac{U}{W}, f\frac{V}{W}). \tag{12}$$

Suppose now that an error is made in determining the displacement (u, v), so that we find (u', v'), where

$$(u', v') = (u, v) + (a, b). \tag{13}$$

Thus, substituting (10) and (11) into (13), we find:

$$\begin{aligned} u' &= f\frac{U}{Z} - x\frac{W}{Z} + a, & (14)\\ v' &= f\frac{V}{Z} - y\frac{W}{Z} + b. & (15) \end{aligned}$$

For purposes of illustration, let us assume that the scene being viewed is coplanar at depth Z with the image plane, and that (a, b) is the same for each image point. We can combine the coordinate independent terms in (14) and (15) to find:

$$\begin{aligned} u' &= f\frac{(U + \delta U)}{Z} - x\frac{W}{Z},\\ v' & \quad f\frac{(V + \delta V)}{Z} - y\frac{W}{Z}, \end{aligned}$$

where the *constants* δU and δV are given by

$$\delta U = aZ/f, \quad \delta V = bZ/f. \tag{16}$$

The displacement field is therefore of the same form as (10,11), and hence there is still an FOE in this case, located (according to (12)) at

$$(x_0', y_0') = (f\frac{U + \delta U}{W}, f\frac{V + \delta V}{W}) = (x_0, y_0) + (\delta x_0, \delta y_0), \tag{17}$$

where

$$(\delta x_0, \delta y_0) = \frac{Z}{W}(a, b). \tag{18}$$

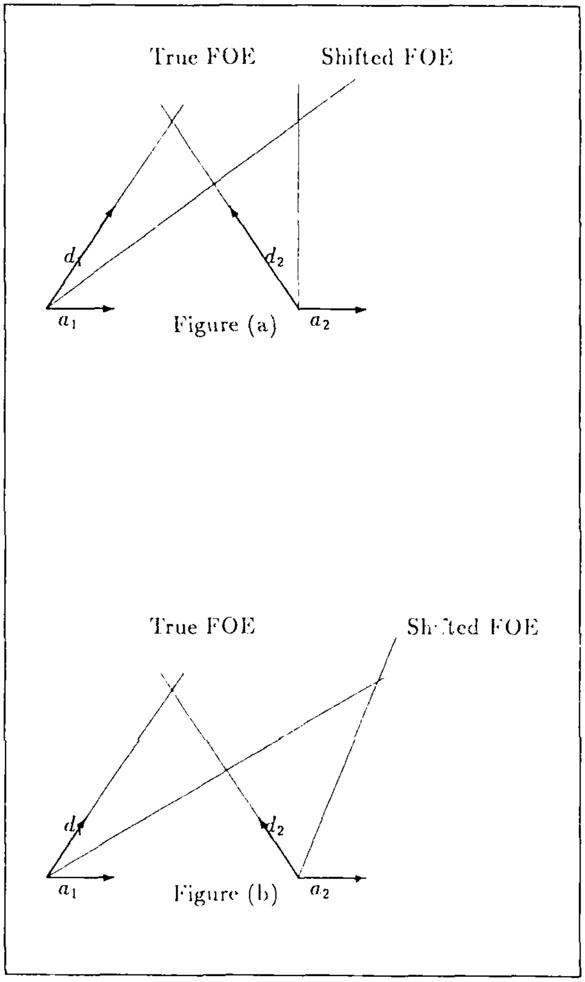
We will call this the "error in the FOE" due to the error (a, b) in the image displacement of the point located at depth Z.

Suppose now that we allow the displacement errors (a, b) in (13) to vary with position in the image plane, and that the environment is no longer coplanar with the image plane. It follows that the "errorful" displacement will no longer be radial, and hence there will not, in general, be an FOE. However, we could still apply our purely translational algorithm to this flow field. How will the "errorful" flow field affect our algorithm?

This is clearly not a question that can be given a general answer. In fact, the only result we can quote is equation (18), which we have proved only in the case of constant depth and constant error (a, b). We can, however, conclude one general rule from (18): for fixed, constant displacement error, distant points (those for which $Z \gg W$) will contribute a larger error to the position of the "FOE" than will nearby points (those for which Z and W are comparable). This is to be expected on intuitive grounds as well, since (generally speaking) distant points will have smaller displacements than nearby points, and so a given displacement error will have a proportionally larger effect on the displacement of distant points than on that of nearby points.

This situation is illustrated in Figure 2. In Figure 2(a) the shift in the FOE is smaller than in Figure 2(b) because the displacement vectors d_1 and d_2 in Figure 2(a) are larger than the corresponding displacement vectors in Figure 2(b). The displacement vectors are larger when the points are closer. Hence, the larger percentage error in displacement for distant points returns a FOE position with greater error.

Figure 2: Shift in the FOE relative to the size of the displacement vectors



We note that no assumption about the source of the displacement error has been made it could be noise in the input image, correspondence error, or anything else. The usual assumption is that the displacement errors are due to random, uncorrelated noise. This would lead to displacement errors which are random in both magnitude and direction, and so on average would not lead to a large error in the position of the FOE. However, if there is a systematic error, for instance when the displacement error is a constant, then the FOE error would be expected to be given by an expression something like equation (18). In the next section, we show that under certain not terribly restrictive conditions rotations can give rise to just such a systematic error.

3.4 Rotations as a Source of Systematic Error

When the scene undergoes a general rigid motion, the displacement field will be given by (5), with translational and rotational components by equations (6)–(9). We can, a la equation (13), consider the rotational components (u_r, v_r) as an error term on the underlying pure translational displacement field (u_t, v_t). From (8) and (9), we see that dis

Table 1: **Rotational contributions to optical flow as a function of position.** Shows the (constant,linear,quadratic) contributions (in pixels) to the x-component of the displacement field at various spatial locations

x(0.54/-0.22/0.19)	x(0.54/-0.22/0.07)	x(0.54/-0.22/0)	x(0.54/-0.22/-0.02)	x(0.54/-0.22/0)
x(0.54/-0.11/0.14)	x(0.54/-0.11/0.05)	x(0.54/-0.11/0)	x(0.54/-0.11/0)	x(0.54/-0.11/0.05)
x(0.54/0/0.09)	x(0.54/0/0.02)	x(0.54/0/0)	x(0.54/0/0.02)	x(0.54/0/0.09)
x(0.54/0.11/0.05)	x(0.54/0.11/0)	x(0.54/0.11/0)	x(0.54/0.11/0.05)	x(0.54/0.11/0.14)
x(0.54/0.22/0)	x(0.54/0.22/-0.02)	x(0.54/0.22/0)	x(0.54/0.22/0.07)	x(0.54/0.22/0.19)

"error" term has the following structure:

$$u_r = \{+Bf\} + \{-Cy\} + \{\frac{Ay+Bx}{f}x\}, \qquad (19)$$

$$v_r = \{-Af\} + \{+Cx\} + \{\frac{Ay+Bx}{f}y\}. \qquad (20)$$

That is, each component has a constant term, a term which varies linearly with image coordinates, and a term which is quadratic in image coordinates. This is illustrated in Figure 3, where we see how the flow deviates more and more from a constant as we approach the periphery of the image, where the linear and quadratic terms become important.

Let us see how the constant, linear, and quadratic contributions compare to one another. We recall from equation (2) that f is the camera focal length in pixels. For instance, for a 256 × 256 image with 45° field of view, f is about 309 pixels. The coordinates x and y, on the other hand, are limited to being less than or equal to $N/2$ in magnitude, which in this case is 128. Thus,

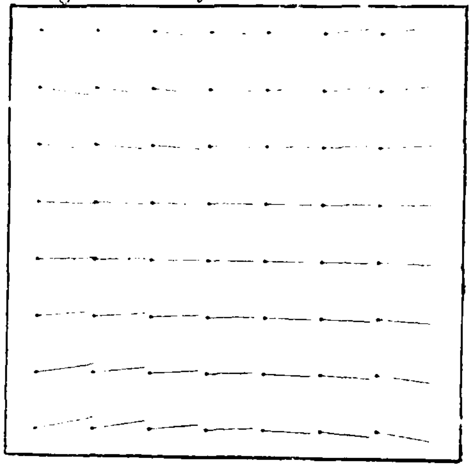
$$|\frac{x}{f}|,|\frac{y}{f}| \le \tan \mathrm{FOV}/2 \sim \frac{128}{309} \sim 0.4. \qquad (21)$$

For rotational angles which are comparable ($A \sim B \sim C$), we see from (19,20) that the order of magnitude of the linear and quadratic terms, with respect to the constant term, is given roughly as

$$\frac{\text{linear}}{\text{constant}} \sim 0.4,$$

$$\frac{\text{quadratic}}{\text{constant}} \sim 2 \cdot (0.4)^2 \sim 0.32.$$

Figure 3: Purely rotational flow fields.



Indeed, these upper limits are reached only at particular regions near the image boundaries. For the greater part of the image surrounding the center, the linear and quadratic terms are *much* smaller than the constant term. This is illustrated in Table 1, where we show the values taken by the constant, linear, and quadratic terms contrib:.tions to u at various points in a 256 × 256 image, with a field of view of 45°, for a rotation of $A \cdot B \cdot C = 0.1°$.

This leads us to make the assumption that the dominant effect of rotations is to add a constant term to the displacement field. We see that this will be a good assumption when:

1. The field of view of the camera is small (say < 45°).
2. The Z component of the rotation is comparable to, or smaller than, the X and Y components of the rotation.

In particular, the assumption of constant displacement error is *not* valid when the primary rotation is around the line of sight (roll) (i.e., when C is much larger than A or B), since the linear term will then be larger than the constant term over a large portion of the image.

3.5 The Effect of Rotation on FOE–error

In this section we combine the results of the previous two sections to find how rotations affect the position of the FOE.

In Section 3.3 we showed that for an environment coplanar with the image plane located at depth Z a constant displacement error leads to a displacement field which is still purely translational, so that an FOE exists. The error in this FOE due to the displacement error is given by equation (18). In Section 3.4 we argued that under certain not terribly restrictive conditions, the dominant effect of rotations was to add a constant error to the displacements, given by the constant term in equations (19,20):

$$(a,b) \quad (Bf, \quad Af). \tag{22}$$

By combining (18) and (22), we find that the X and Y rotations give an FOE error (in pixels) of

$$(\delta x_0, \delta y_0) \quad \frac{Zf}{W}(B, \quad A). \tag{23}$$

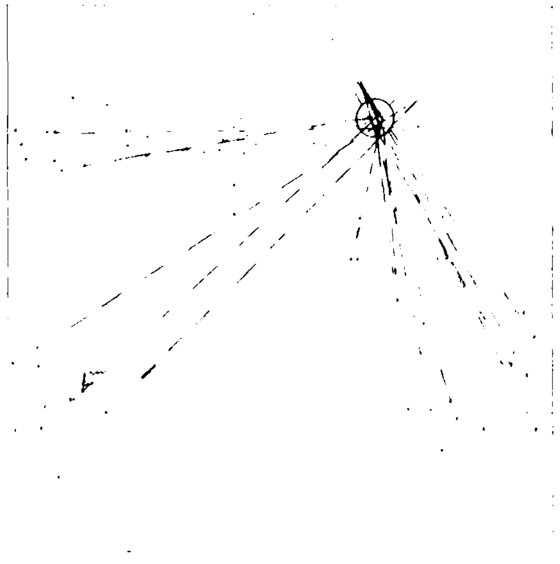
To see how large an effect this is, let the FOV for a 256 × 256 image be 45°, and let $B \quad A$. In Table 2, we show $|\delta x_0|$ for various values of Z/W and of B. The first column gives the rotational angle, the second the displacement error (in pixels) for each angle, and the remaining columns give the FOE error (in pixels) for each angle.

Table 2: **Error in FOE (in pixels) with rotational angle and Z/W**

rotational	corresponding	Z/W					
error, B	disp.,Bf	5	10	20	30	50	100
0.1deg.	0.5 pixels	2.5	5	10	15	25	50
0.2deg.	1.0 pixels	5	10	20	30	50	100
0.3deg.	1.5 pixels	7.5	15	30	45	75	150
0.4deg.	2.0 pixels	10	20	40	60	100	200

Of course, it is exceedingly rare that the environment and the image are coplanar, so these results cannot be strictly applied. However, we can draw a few general conclusions from this analysis. The first conclusion is that when an image sequence contains even a small amount

Figure 4: Estimated FOE obtained by extending the larger flow vectors.



of rotation, the error in the FOE can be large. This error will be larger when the environmental points used to find the FOE are more distant. It would therefore be advantageous to weight nearby environmental points more than distant ones. We note that the fundamental reason for this is that nearby points will in general have larger interframe displacements than distant points. Thus, to minimize the FOE error we should give points with large displacements greater weight than those with small displacements (eg., Figure 4 shows the larger flow vectors obtained from frame pair 9-11 of Figure 5 extended to give an approximate FOE).

The second conclusion is that since most schemes round off displacements to the nearest half-pixel, the first row in Table 2 gives the worst-case error in finding the FOE for such schemes. In general, such rounding-off will give randomly-distributed displacement errors, which should not have much effect on the FOE. The worst-case error assumes that the displacement errors are not random, and hence do not cancel out.

4 Removing Rotation without Analyzing General Motion

We have shown in the previous sections that if an algorithm which assumes purely translational motion is applied to an image sequence which has even a small amount of interframe camera rotation, the "FOE" found by the algorithm can be far from the FOE that would be found if no rotation were present. Thus even small rotations can seriously degrade the performance of a purely translational algorithm.

One possible solution to this difficulty would be to somehow remove the (small) rotational component of the camera motion, producing a so-called *registered* image sequence in which the inter-frame camera motion is purely translational. This possibility has recently been investigated by Pavlin and Snyder [14]. In the next sections we discuss this registration scheme, its limitations, and some experimental results from using it.

4.1 The Registration Algorithm

Consider a dynamic image sequence $\{F_1, \ldots, F_n\}$ in which the camera is allowed to have general inter-frame motion. If we wish to run a purely translational motion algorithm on this sequence we will, as just discussed, obtain unreliable results in general because of the existence of small rotational components to the inter frame motion. Suppose, however, that we could somehow find the rotational components of the inter frame motion. We could then subtract out this motion from the second of each image pair, obtaining thereby a *new* image sequence in which the inter frame camera motion is purely translational. This new image sequence $\{F_1^{\mathrm{reg}}, \ldots, F_n^{\mathrm{reg}}\}$ will be called a *registered* image sequence. The translational motion algorithm could then be safely applied to this registered sequence.

Unfortunately, for reasons given in [14], the construction of the registered image sequence is in principle impossible, but an approximation to a registered image sequence can be constructed as follows (see [14] for more details, justifications, etc.). The key idea behind the registration algorithm is that (infinitely) distant points move, from frame to frame, only because of rotations (this is a trivial consequence of the general optical flow equations). Furthermore, if these points are not too far from the FOE, the terms in (u_r, v_r) which are quadratic in image coordinates will be negligible compared to the constant and linear terms, and hence may be ignored. We will call the points satisfying these conditions the set of *registration* points. As a consequence, the displacement field for such registration points is extremely simple:

$$\begin{aligned} u = u_r &= \{Bf\} + \{-Cy\} \\ v = v_r &= \{-Af\} + \{+Cx\}. \end{aligned}$$

We note that this displacement field is just a rotation by an angle C around the origin, followed by a translation $(Bf, -Af)$ in the image plane, the combination of which is just a general rigid transformation of the image plane. If we now consider the set of all such registration points, it is clear from the displacement field above that these registration points move from one image to the next as a rigid body.

The idea in [14] is to use an optical flow algorithm, such as that of Anandan [8], to find point correspondences between successive frames of the unregistered image sequence. Picking a certain set of such points and considering it as a (two-dimensional) rigid body, one can define the center of mass of this set of points, and the principal axes of its inertia tensor. The rotation angles A and B can then be found by finding the displacement of the center of mass of this rigid body between the frames in question, and the rotation angle C can be found by finding the rotation of the principal axes between the two frames. Thus, the inter-frame rotational component of the camera's motion can be found.

Once these rotations have been found, then the registered image can be obtained by subtracting off the rotational component of the displacement at each pixel. Unfortunately, this is a nonlinear transformation, and hence will be computationally expensive. As a consequence, the registered image is found by linearizing the rotational component, that is by performing the inverse of the rigid motion, i.e., translate the second image by $(-Bf, +Af)$, then rotate it by $-C$ around the origin.

In practice, the registration algorithm of [14] translates the second image by $(-Bf, +Af)$, rounded off to the nearest whole pixel, and does not perform the rotation by $-C$ around the origin. The former approximation is made in the interests of computational efficiency, since sub-pixel interpolation would be expensive. The latter approximation is made since the quantization grid of a rotatated image does not coincide with the initial quantization grid, so a complicated sort of interpolation would be necessary.

4.2 Difficulties with the Registration Algorithm

There are a number of difficulties with the registration algorithm described above, but the primary one is this: How do you find the set of registration points, those environmental points which are infinitely distant from the image plane, yet whose image projections are close to the FOE ?

The answer, of course, is that it is difficult (if not impossible) to know these things a priori. To be sure, in some domains one might be able to specify a region of the image in which distant points might be expected to be found, but in general, one may not know this. The second point we want to make is that genuinely "infinitely distant" points are rare, so the question then becomes one of whether distant points are distant "enough."

The basic idea is that an environmental point is distant "enough" if its image displacement due to translation is much smaller than its image displacement due to rotation. Only then can one say that the bulk of the displacement is due to rotation. We can easily derive a condition that expresses this. From the time-adjacency relation (1), we see that the displacement d due to translation is just

$$d = D\frac{W}{Z}. \tag{24}$$

For image points near the center of the image, their displacement is, as we have shown previously, of order Af (we assume that $B = C = 0$ for simplicity). Combining these two results, we see that the translational displacement (d) will be much smaller than the rotational displacement (Af) if

$$D\frac{W}{Z} = d \ll Af \rightarrow \frac{Z}{W} \gg \frac{Z_0}{W} \equiv \frac{D}{Af}. \tag{25}$$

This therefore implies that a point is "distant" enough only if Z/W is much larger than Z_0/W, where

$$\frac{Z_0}{W} = \frac{D}{Af}. \tag{26}$$

In Table 3 we list the values that the quantity Z_0/W takes for various values of rotation Af and various distances D from the "FOE" (in pixels). For example, with a rotation of 0.2 degrees and a point at a distance of 25 pixels from the "FOE", $\frac{Z_0}{W}$ is 23. Therefore in our experiment with a translation of four feet between frames (i.e. W = 4 feet) a point must be greater than 92 feet for the approximation to hold. For other rotations and distances from the FOE, the values required can be considerably larger. We see that except for quite large angles (of the order of several degrees), and image points quite near the FOE, "distant" points are unlikely to be found in typical images.

Table 3: **Minimal distances Z_0/W for which points can be considered "distant".** Only points for which $Z/W \gg Z_0/W$ can be considered "distant" (see text for an explanation)

Rotation A	Af	Distance D in pixels			
(degrees)	(pixels)	25	50	75	100
0.1	0.5	50	100	150	200
0.2	1.1	23	45	68	91
0.5	2.7	9.3	18	28	37
1.0	5.4	4.6	9.3	14	19
2.0	10.8	2.3	4.6	6.9	9.3
5.0	27.0	0.9	1.9	2.8	3.7

Additional difficulties with the registration algorithm are of an implementational nature. The approximation of the (nonlinear) rotational displacement by its linear terms will introduce errors, especially in the displacements of peripheral points (which are precisely those with the greatest displacement in general, and which, as we saw in the previous section, are therefore those which should receive the greatest weight in the finding of the FOE). The rounding off of the rigid displacement ($-Bf$, $+Af$) to the nearest pixel could introduce errors of up to 100 % in the removal of the rotational parameters (for instance, if $Bf = 0.51$ pix, it would be rounded off to 1 pix). The neglect of the "roll" term by not performing the rotation about the origin by $-C$ could introduce severe difficulties into an FOE finding algorithm.

There are, as mentioned in [14], several ways of eliminating some of the problems with the registration algorithm. The most severe problem with the algorithm, the difficulty of ensuring that the registration points are distant points, could be minimized if, for instance, motion were integrated with stereo, or with some sort of active sensing. The latter modules could then be used to select regions in the image which contain distant points.

Other difficulties, which are essentially related to the finite resolution of the image plane, would be minimized by working with high resolution images. In this regard, it should be noted that aside from the correspondence component, the complexity of the registration algorithm is independent of the resolution.

In order to see how the the combination of the registration algorithm and the FOE finding algorithm performs, we ran the combined algorithm on several frames from the image sequence described in the next Section, and illustrated in Figure 5. The position found for the

Table 4: **Recovered FOE's with and without Registration.** The image size is 256 by 256 with (0,0) at the centre. The estimated FOE is around (27,80).

Frames	FOE (unregistered frames)	Registration Correction in pixels (x,y)	FOE (registered frames)
1–3	(177,149)	(-0.1, 1.9)	(44,135)
3–5	(162,47)	(-0.8, 1.1)	(261,163)
5–7	(226,142)	(0.0, 1.6)	(36,111)
7–9	(244,109)	(0.0, 1.6)	(282,126)
9–11	(243,109)	(0.1, 1.1)	(51,76)

FOE from each pair of frames is shown in Table 4. The "correct" FOE position (found from known camera parameters and measured vehicle motion) should be somewhere around $(x_0, y_0) = (27, 80)$, where the origin is in the center of the image, and the distances are given in pixels. We see that some of the values found for the FOEs are not too bad, while others are completely erroneous (although even the best in this experiment would not be sufficient for depth extraction). We conclude that this combination of algorithms does not generally give reliable FOEs.

The solution to our problem would seem to be an algorithm that finds both the rotational and translational motion parameters (i.e the FOE) at the same time. That is, we should consider a general motion algorithm. In the next section we consider such an algorithm constructed from the work of Anandan and Adiv at UMass, and show that promising results are obtained on the CMU NAVLAB image sequence.

5 Depth from a General Motion Algorithm

In this section we review a general motion algorithm to estimate the motion parameters and depth as implemented at the University of Massachusetts. We then show the results of applying this algorithm to a real image sequence. Finally, we compare this algorithm to the approximate translational motion scheme discussed in the previous sections.

5.1 The General Motion Algorithm

The general motion algorithm implemented at the University of Massachusetts to compute 3-D motion parameters and depth is based on the work of Anandan [8,9,10] and Adiv [12]. We discuss the two phases of the algorithm below.

5.1.1 Displacement Field Determination

In the first phase Anandan's [8] algorithm is used for determining displacement fields. It operates on a pair of images and uses a hierarchical correlation matching approach. A multi-resolution, multiple spatial frequency channel representation of the images is used for efficient computation and accurate determination of displacement fields. Confidence measures indicating the reliability of each displacement vector are provided. A smoothness constraint is then used to correct unreliable vectors based on the reliable ones.

5.1.2 Depth and Motion Parameter Estimation

In the second phase the flow field, along with associated confidences, found via Anandan's algorithm is taken as input to Adiv's [11,12] algorithm. This algorithm consists of two steps, which we now describe.

In the first step the flow field is segmented into connected sets of flow vectors where each set is consistent with the rigid motion of an approximately planar patch. The segmentation is based on a modified version of the generalized Hough transform, with displacement vectors voting for the motion parameters. It is expected that each segment would correspond to the motion of a portion of only one rigid object. The segmentation approach makes it possible to deal with independently moving objects. When there are no independently moving objects, this stage of segmenting the flow field is unnecessary and the entire static environment can be viewed as a single rigid object in the second phase of the algorithm.

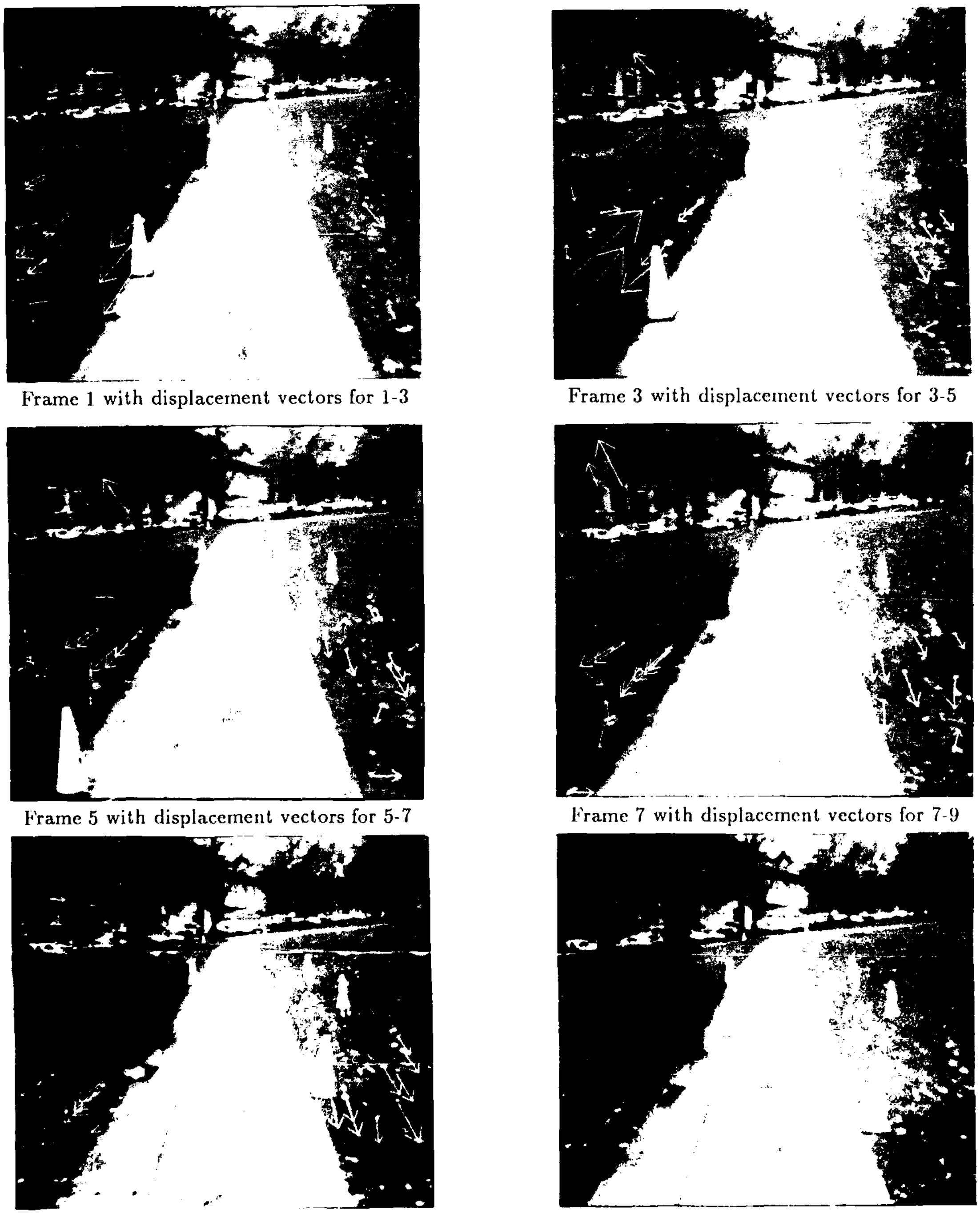
In the second step, the segments found in the first step are *grouped* together under the hypothesis that they have been induced by a single rigidly moving object (i.e. the planar surface assumption is dropped). This is done by computing the optimal motion parameters and related error measure for each segment by employing a least-squares approach that minimizes the deviation between the measured flow fields and that predicted from the estimated motion and structure. For each hypothesized FOE the optimal rotation parameters and the related error value are computed. A multiresolution discrete sampling technique is used to compute the minimum value of the resulting error function. *This step essentially involves grouping segments of the flow field which are consistent with the same motion parameters.*

Following the computation of 3-D motion parameters the depth can easily be computed if the total translation between frames is known.

5.2 Experimental Domain

In an effort to determine *ground truth for the experiments* a sequence of 20 images were taken using the NAVLAB at Carnegie-Mellon University. The field of view of the

Figure 5: The sequence of image frames.



Frame 1 with displacement vectors for 1-3

Frame 3 with displacement vectors for 3-5

Frame 5 with displacement vectors for 5-7

Frame 7 with displacement vectors for 7-9

Frame 9 with displacement vectors for 9-11

Frame 11

Table 5: **Depth Values of some points over a sequence of frames using the General Motion Algorithm.** The two tables used 100 and 200 points respectively. Depths are in feet. * and @ indicate respectively that the point was not among the top 100 or top 200 Moravec points. ** indicates that the point is absent in the image-pair.

		1-3	1-3	3-5	3-5	5-7	5-7	7-9	7-9	9-11	9-11
Object	pts.	Exptl	True	Exptl	True	Exptl	True	Exptl	True	Exptl	True
cone1	1	65.7	76	58.3	72	61.7	68	50.3	64	61.2	60
	2	66.9	76	67.7	72	60.5	68	63.6	64	59.6	60
cone2	3	61.4	76	67.2	72	65.1	68	63.0	64	63.9	60
	4	60.8	76	82.3	72	56.2	68	61.7	64	61.7	60
cone3	5	50.2	56	59.2	52	46.3	48	40.8	44	38.4	40
	6	51.1	56	49.6	52	46.1	48	41.0	44	38.5	40
cone4	7	50.3	56	53.8	52	44.4	48	35.9	44	37.9	40
	8	46.3	56	53.3	52	47.6	48	41.8	44	39.8	40
can	9	44.1	46	44.4	42	47.6	38	41.8	34	39.8	30
	10	*	46	*	42	*	38	*	34	*	30
	11	*	46	*	42	*	38	*	34	*	30
cone5	12	31.0	36	32.2	32	26.0	28	22.0	24	20.0	20
	13	31.1	36	31.1	32	26.3	28	22.5	24	20.8	20
	14	31.9	36	30.9	32	28.5	28	21.9	24	20.5	20
cone6	15	18.1	21	*	17	**	**	**	**	**	**
	16	18.4	21	*	17	**	**	**	**	**	**
	17	18.9	21	-29	17	**	**	**	**	**	**
	18	18.6	21	-42	17	**	**	**	**	**	**

		1-3	1-3	3-5	3-5	5-7	5-7	7-9	7-9	9-11	9-11
Object	pts.	Exptl	True	Exptl	True	Exptl	True	Exptl	True	Exptl	True
can	9	38.1	46	43.3	42	36.5	38	32.2	34	29.6	30
	10	40.4	46	@	42	@	38	30.3	34	32.4	30
	11	44.2	46	42.5	42	@	38	32.7	34	@	30

camera was measured, as was the inclination of the camera axis from the road plane. Traffic cones were placed at measured locations in the environment, and the actual interframe translation of the vehicle was measured and marked on the roadway, so that ground truth depth data could be reasonably precisely obtained. We obtained a sequence of about 20 usable images some of which are shown in Figure 5.

5.3 Experimental Results

Experiments to determine depth and motion parameters were conducted using the general motion algorithm described above. To speed up computation of Anandan's algorithm, "interesting" or distinctive points were found via the Moravec operator in the images and flow fields determined only at these locations. Two sets of experiments were performed - one using 100 points and the other using 200 points). The flow fields are shown in Figure 5. False matches are sometimes produced, particularly near the boundary of the image due to feature points moving out of the image. This is particularly noticeable in the frame-pairs (3-5) and (5-7).

Adiv's algorithm was run on the flow fields produced by Anandan's algorithm. As mentioned earlier the segmentation part of Adiv's algorithm was unnecessary because there is in effect only one moving object. The motion parameters produced by Adiv's algorithm are shown in Table 6.

The translation results are all the same except for the frame pair 3-5 run with 200 points. The result suggests that Adiv's algorithm finds the translation vector fairly well. The less accurate results for frame pair 3-5 run with 200 points are because of false matches at the boundary of the image aggravated by choosing points which are not distinctive.

The results also indicate an approximate constant rotation of about 0.4 to 0.5 degrees about the Y-axis, a small and varying component about the X-axis, and a random rotation about the Z-axis. This is consistent with

Table 6: **Motion Parameters obtained through the General Motion Algorithm.** The frame pairs are at 4 ft. intervals. The results have been tabulated for 100 Moravec points and 200 Moravec points. (U,V,W) is the unit translational vector and (A,B,C) are the rotational components in degrees

100pts	1–3	3–5	5–7	7–9	9–11
U	-0.09	-0.09	-0.09	-0.09	-0.09
V	-0.25	-0.25	-0.25	0.25	-0.25
W	-0.96	-0.96	-0.96	-0.96	-0.96
A	-0.19	0.17	-0.10	-0.04	-0.03
B	0.39	0.56	0.53	0.49	0.43
C	-0.30	0.01	0.07	0.06	0.28

200pts	1–3	3–5	5–7	7–9	9–11
U	-0.09	-0.16	-0.09	-0.09	-0.09
V	-0.25	-0.21	-0.25	-0.25	-0.25
W	-0.96	-0.96	-0.96	-0.96	-0.96
A	-.19	0.11	-0.10	-0.03	0.03
B	0.41	0.17	0.53	0.49	0.43
C	-0.22	-0.52	0.10	0.07	0.31

1. (x-axis): a road surface which deviates very slightly from being planar either because of bumps or the surface itself being non-planar.

2. (y-axis): a small drift of the vehicle to the right.

3. (z-axis): some random motion probably due to the vehicle roll.

However , we do not have any measurements of the rotational components to verify how correct the derived rotational parameters are.

The depth results (for the set with 100 points) are shown in Table 5 for the obstacles on the road (the points are labelled in Figure 6). Since several points are missing in the case of the can,for the can we also show depth values when 200 interest points were used. In general, these are fairly good when compared to other attempts at the recovery of the depth of points (average error roughly 15%). The results improve for some of the later framepairs. Occasionally absurd depth values are returned because of false matches; this is particularly noticeable for cone 6 when it is very close to the edge of the image. For points which are at a distance, the depths returned are usually reasonable with some absurd values returned because occlusion causes false matches. The depth values obtained by using 200 interest points show small variations from the ones in table 5. Since in most cases the translation parameters found and the displacements of the points are the same, this suggests that depth computation is very sensitive to the accuracy with which the rotational parameters are computed.

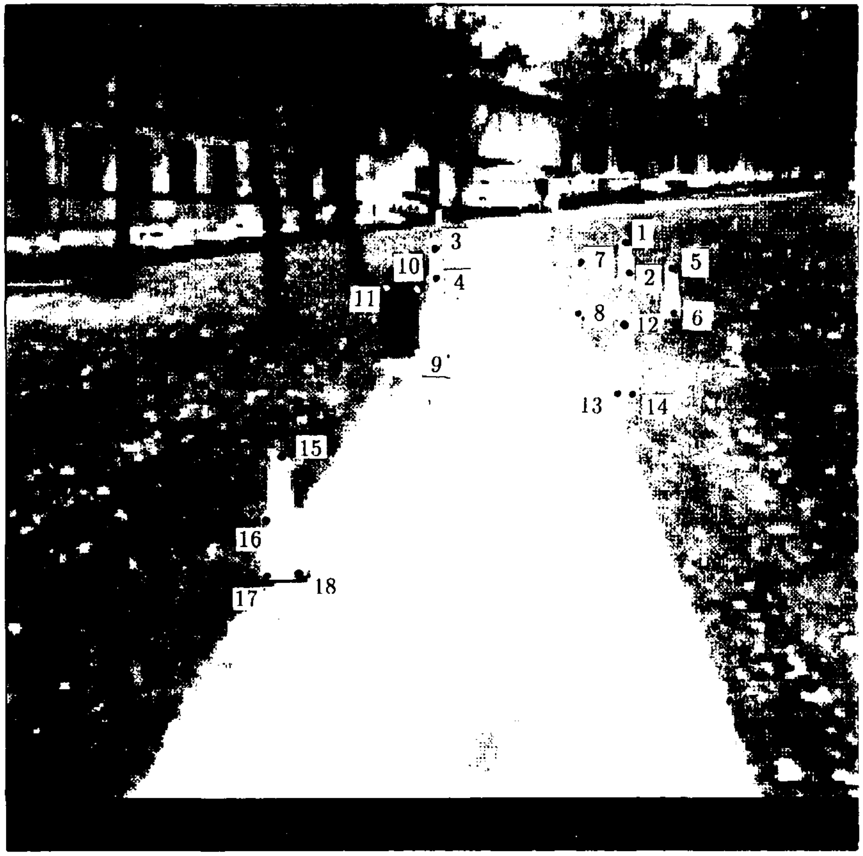
It should be noted that in view of the results reported by earlier researchers regarding structure computations this appears to be a very promising result. It also appears as if this technique of solving the motion problem is superior to the methodologies which rely on approximate translational algorithms. It should also be noted that the displacement vectors found by Anandan's algorithm are indeed remarkable as can be seen from the displacement vectors shown in the images at the end of this paper.

6 Conclusion

It is our contention that the computation of the depth of points from approximately known translational motion is difficult to determine in the presence of even small rotations. If the rotational effects are not removed, large errors in the location of the FOE can occur, and the effect upon depth recovery can be disastrous. One scheme to eliminate small rotations by using distant points, which would have primarily rotational displacements was not effective. While each of the algorithms may work well in isolation from each other, a global system is much more difficult to construct. The cumulative errors from the subsystems tend to propagate to give disastrous results on many occasions. The algorithms developed by Anandan for computing displacements, and Adiv for computing the parameters of general sensor motion and depth, were applied to the same image sequence. The results appear to be quite good, although ground truth for sensor motion was not available. The extracted translational motion across five pairs of frames (20 feet) in the environment was constant, while the rotation was computed to be small; (we note that no ground truth was available for the rotational components). The patterns of these rotations seems quite reasonable for the context of the vehicle moving on a planar path. The extraction of depth from many of the objects was reasonably good, and sometimes surprisingly accurate at distances of 40 feet or greater.

The use of a gyroscopically stabilized platform would allow the simpler translational algorithms to be applied in extracting the depth of points. Translational motion processing should be computationally more efficient and effective. As an alternative to the algorithms described in this paper, a variety of other methods have been proposed to determine depth and motion parameter from image sequences. Accurate depth determination through methods based only on stereo are probably difficult to apply from a moving vehicle because of the rather small baseline. However, a combination of stereo and motion has been proposed [23,24,25], including recent research from our group and presented in this volume [26], in a form that might provide additional constraints to be effective. Alternatively the detection of features like lines might be more effective in depth computation. Current research (also presented in this volume) describe token-based techniques for computing correspondence of linear structures

Figure 6: The interest points marked on frame 1 of the image sequence



over time in order to extract depth from looming [27,28]. Initial experimental results have shown cases where depth extraction has been quite accurate.

In summary, there are currently no practical motion systems providing robust and accurate determination of depth in real scenes. However, there are promising directions actively being explored. Nevertheless, it will require considerable effort to develop a working system which will perform robustly in view of many of the problems that occur in real world situations.

7 Acknowledgements

We would like to thank Harpreet Sawhney, Rakesh Kumar and Yigal Gur for help with different aspects of the experiments.

This research was supported in part by the DARPA and US Army ETL under contract number DACA76-85-C-0008, and DARPA and RADC under contract number F30602-87-C-0140. Part of the research is also supported by the National Science Foundation under grant number DCR-8500332.

References

[1] D. T. Lawton. *Processing Dynamic Image Sequences from a Moving Sensor.* PhD thesis, University of Massachusetts at Amherst, 1984. COINS TR 84-05.

[2] D. T. Lawton. Processing translational motion sequences. *Computer Graphics and Image Processing*, 22:116–144, 1983.

[3] I. Pavlin, E. Riseman, and A. Hanson. *Translational Motion Algorithm with Global Feature Constraints.* Technical Report COINS TR 86-58, University of Massachusetts at Amherst, 1986.

[4] I. Pavlin, E. Riseman, and A. Hanson. Analysis of an algorithm for detection of translational motion. *Proceedings of DARPA Image Understanding Workshop*, 388–398, December 1985.

[5] M. A. Snyder. Uncertainty analysis in image measurements. *Proceedings of the DARPA Image Understanding Workshop, Los Angeles, California*, February 1987.

[6] S. Bharwani, E. Riseman, and A. Hanson. Refinement of environmental depth maps over multiple frames. *Proceedings of the IEEE Workshop on Motion: Representation and Analysis*, 73–80, 1986. Also in *Proceedings of the DARPA Image Understanding Workshop*, December 1986.

[7] Ronald C. Arkin. *Towards Cosmopolitan Robots: Intelligent Navigation in Extended Man-Made Environments.*

PhD thesis, University of Massachusetts at Amherst, 1987. COINS TR 87-80 , page 304.

[8] P. Anandan. A unified perspective on computational techniques for the measurement of visual motion. *IEEE First International Conference on Computer Vision*, 219–230, 1987.

[9] P. Anandan. *Measuring Visual Motion from Image Sequences.* PhD thesis, University of Massachusetts at Amherst, 1987. COINS TR 87-21.

[10] P. Anandan and R. Weiss. Introducing a smoothness constraint in a matching approach for the computation of optical flow fields. *Proceedings of the IEEE Third Workshop on Computer Vision: Representation and Control, Bellaire, Michigan*, October 1985.

[11] Gilad Adiv. *Interpreting Optical Flow.* PhD thesis, University of Massachusetts at Amherst, 1985. COINS TR 85-35.

[12] Gilad Adiv. Determining three-dimensional motion and structure from optical flow generated by several moving objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 7(4):384-401, July 1985.

[13] C. Thorpe, S. Shafer, T. Kanade, and the members of the Strategic Computing Vision Lab. Vision and navigation for the carnegie mellon navlab. *Proceedings DARPA Image Understanding Workshop*, 143–152, February 1987.

[14] I. Pavlin and M. Snyder. The problem of registration of dynamic image sequences. 1987. Internal Memo, Computer Science Department, University of Massachusetts at Amherst.

[15] R. Y. Tsai and T. S. Huang. *Three Dimensional Motion and Structure from Image Sequences.* New York: Springer-Verlag, 1983.

[16] J. L. Barron, A. D. Jepson, and J. K. Tsotsos. The sensistivity of motion and structure computations. *Proceedings sixth national conference on Artificial Intelligence*, 700–705, 1987.

[17] J. Q. Fang and T.S. Huang. Solving three dimensional small-rotation motion equations. *Proceedings Computer Vision and Pattern Recognition*, 253–258, June 1983.

[18] R. Y. Tsai and T. S. Huang. Uniqueness and estimation of 3-d motion parameters and surface structures of rigid objects. *Image Understanding 1984*, 135–171, 1984.

[19] D. H. Ballard and C. M. Brown. *Computer Vision.* Prentice-Hall Inc., 1982.

[20] A. Bandopadhay and R.Dutta. Measuring image motion in dynamic images. *Proceedings of the IEEE Conference on Motion: Representation and Analysis*, 67–72, May 1986.

[21] A. Bandopadhay and R. Dutta. Measuring motion in dynamic images: a clustering approach. *Proceedings Sixth Canadian Conference on Artificial Intelligence*, May 1986.

[22] B. K. P. Horn. Motion fields are hardly ever ambiguous. *International Journal of Computer Vision*, 1(3), October 1987.

[23] A. M. Waxman and J. H. Duncan. Binocular image flows - steps towards stereo-motion fusion. *IEFE Transactions on Pattern Analysis and Machine Intelligence*, (vol 8 no 6), 1986.

[24] A. M. Waxman and S. Sinha. Dynamic stereo: passive ranging to moving objects from relative image flows. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (vol 8 no 4), 1986.

[25] M. Jenkin and J. K. Tsotsos. Applying temporal constraints to the dynamic stereo problem. *Computer Vision Graphics and Image Processing*, 1986.

[26] Poornima Balasubramanyam and M. A. Snyder. Computation of motion in depth parameters: a first step in stereoscopic motion interpretation. *Proceedings DARPA Image Understanding Workshop*, 1988. Cambridge, Massachusetts.

[27] Lance R. Williams and Allen R. Hanson. Translating optical flow into token matches. *Proceedings DARPA Image Understanding Workshop*, 1988. Cambridge, Massachusetts.

[28] Lance R. Williams and Allen R. Hanson. Depth from looming structure. *Proceedings DARPA Image Understanding Workshop*, 1988. Cambridge, Massachusetts.

A Real Time Hierarchical Model for Optic Flow Determination Via Spatiotemporal Frequency Channels

Ajit Singh **Peter K. Allen**

Department of Computer Science, Columbia University

Abstract

The recent work on perception and measurement of visual motion has consistently advocated the use of hierarchical representation and analysis. In most of the practical applications of motion perception, such as robotics, object tracking in space applications, etc., it is absolutely necessary to be able to construct and process these hierarchical image representations in real time, and this calls for an inherently parallel implementation. In this paper, first we briefly review the past work on motion perception that makes use of hierarchical models and identify the need for their real time implementation. We then describe a new hierarchical model for extraction of one form of motion information from an image sequence, namely optic-flow, based on the recently proposed spatiotemporal frequency approach. We then identify the inherent parallelism in the model and show how to exploit that to implement the model in real-time on the PIPE image processing architecture. In the implementation, we construct the underlying image pyramids and compute the optic flow in real time. Finally we present the results of the PIPE implementation. They are also available on a video-tape to display the real-time nature of the computations.

1 Introduction

Many basic image processing and scene analysis operations can be performed more efficiently and robustly using multiresolution representations. Because of their hierarchical organization and local interconnection patterns, they form the basis of very simple and efficient algorithms for low-level image processing operations and thus render substantial robustness to the higher-level algorithms via the coarse-to-fine strategies. Recent studies in psychology reveal that pyramid is a good model for early stages of human vision. Several such multi-resolution representations have been described in literature [Bur84], [Mal87], [Cro84], [Shn84]. Our interest in pyramid-representations has been because of their promise in the area of detection and measurement of visual motion in the form of *optic flow* [Wax84]. Optic flow depicts the 2-D projections of the 3-D scene velocities on the projection surface. Recent developments in optic flow computation have begun to identify the potential of hierachical representations and processing. Glazer [Gla87] extensively investigated the idea of using hierarchical schemes for motion measurement. Recent work by Anandan [Ana86c] uses Burt's [Bur84] Laplacian pyramid for hierarchical correlation based matching to compute the optic-flow over two successive image frames. Heeger [Hee87b], in his recent work on optic flow using spatiotemporal filtering alludes to the possibility of using multi-resolution schemes for more robust results.

The techniques described in the literature for determination of optic-flow [Sin87c] lie in the following three basic categories: ***intensity gradient and difference based methods*** such as the classic works by Horn and Schunck [HS81], Thompson and Barnard [TB87] etc.; ***intensity correlation based methods*** such as the recently proposed model by Anandan [Ana86a] and; ***spatiotemporal frequency based methods*** that formulate the problem of optic-flow computation as that of detecting oriented edges in space-time, such as recent works by Adelson and Bergen [AB85], Heeger [Hee87b].

The objective of this paper is three fold. Firstly, we show, with examples from past research that there is a potential promise of hierarchical schemes in at least the *last two* of the abovementioned three categories of methods for optic-flow computation. Secondly we propose a new hierarchical computational model that lies in the third category described above. It uses a simple pyramid scheme which we call the ***pyramid of oriented edges***, or a POE, to decompose the incoming image sequence into oriented spatiotemporal frequency channels that can be used to determine optic flow. The POE is described as a logical extension of Burt's pyramid. We also compare our POE to the functionally (but not computationally) similar Mallat's pyramid [Mal87], that has shown promise in optic flow computation [Hee87a].Thirdly, we identify that for these approaches to be practically applicable, it is of utmost importance to be able to construct the underlying image structures, namely image pyramids, and compute optic flow in real time. The practical issues of real time computation of optic-flow have not been investigated in the past research. We show that by exploiting the inherent ***spatial and temporal parallelism*** in the model, we can come up with ***pixel parallel algorithms*** that can be conveniently implemented on the state of the art image processing architectures such as the PIPE [The87]. We show results of real-time implementation of the POE as well as the optic-flow derived from it, on the PIPE system. For the purpose of comparison, we also show implementation of Mallat's pyramid discussed above.

The organization of this paper is as follows. The three objectives described above are covered in Sections 2, 3 and 4 respectively. Section 5 summarizes this work and enumerates the areas in which there is scope for further work along these lines. It particularly emphasizes the need for, and a possible approach to, the *integration* of the *two* optic-flow models mentioned above.

2 Optic Flow Determination - A Case for Hierarchical Models

In the previous section we had alluded to the fact that hierarchical schemes had a potential application to at least the last two of these three categories namely ***intensity correlation based methods*** and ***spatiotemporal frequency based methods***. In this section we elaborate on this observation. The purpose is ***not merely to provide a review***. The purpose is to illustrate two different approaches whose computational model can be built around a ***common image structure***, namely, ***some*** image pyramid. This common image structure can be used as the tool to ***integrate*** the two approaches to get a more robust model. We discuss the issue of integration as a part of our ongoing work in the concluding remarks at the end of the paper.

Intensity-correlation based techniques essentially use two successive images from an image sequence. They use windows of defined size around all (or some selected) pixels in the first image to look for a pixel in the second image, which has a similar intensity structure in the window around it. Vaguely speaking these techniques search for "matching" pixels. The matching process suffers from a combinatorial explosion.The effectiveness of correlation based techniques can be improved by using a hierarchical search strategy to avert the combinatorial explosion. This technique was successfully used by Anandan [Ana86b]. He used Burt's Laplacian pyramid in the matching process. *Spatiotemporal Frequency Based Methods* have recently shown a significant promise in the problem of motion perception [AB85], [AA85]. Heeger [Hee87b], in his recent work based on spatiotemporal-frequency approach, made use of gabor filters for determination of optic-flow. He alluded in his work that a multiresolution scheme such as Burt's pyramid can be used to construct families of gabor filters that will give a more robust estimate of optic-flow as compared to the current version of his model, that uses only one such family of gabor filters. He has also indicated [Hee87a] the usefulness of another pyramid representation, namely Mallat's pyramid [Mal87], in implementing his model. A very brief functional description of Mallat's pyramid and its applicability to optic-flow determination will be given elsewhere in this paper. Rather than reviewing Heeger's model in order to show the promise of hierarchical schemes to spatiotemporal frequency based methods, we will defer this to the next section, where, after a systematic description of motion in spatiotemporal frequency domain, we will actually describe a new hierarchical computational model that can be used to compute optic-flow based on spatiotemporal frequency channels.

3 Pyramid of Oriented Edges: A Hierarchical Model for Spatiotemporal Frequency Based Optic Flow Computation

3.1 Motion In Frequency Domain

An image is essentially a two dimensional signal. Likewise, a sequence of images is a three dimensional signal. It is easy to see [WA83], [AA85] that the spatial and temporal frequencies are related as:

$$\omega_t = \omega_x U_x + \omega_y U_y \text{..........}(Eqn.1)$$

Here, ω_t is the temporal frequency and ω_x and ω_y are the spatial frenuencies. in x and y direction respectively, in the neighborhood of a point in the image. Also, U_x and U_y are the image velocity (optic flow) components in x and y direction. Eqn. 1 suggests a technique for recovering the image velocity. The mechanism would have to "detect" spatial and temporal frequencies in the sequence of images, making use of small spatial and temporal neighborhoods and use them to compute the image velocities. In order to further fix this idea we show how this follows directly from the conventional space-time formulation of motion.

3.2 Space-Time Domain v/s Spatiotemporal Frequency Domain: What's the Connection ?

A very concise, but perhaps not very comprehensive way to answer this question is tc use Fourier transform based analysis. We will, on the other hand, take a a very qualitative approach to investigate this issue. For this purpose, we will make extensive use of the example of vertical bar moving in a horizontal direction used by Adelson and Bergen [AB85]. This example consists of a vertical bar moving to right as shown in Figure 1a (after Adelson). Figure 1b shows a three dimensional spatiotemporal diagram of the moving bar, that shows up as a slanted slab in space-time. Figure 1c shows a cross-section of parallel to the x-t plane (the y dimension, containing no useful information, can be dropped). Figure 2a (also after Adelson) shows the same x-t slice for various values of the velocity. It can be seen that the velocity can be measured by measuring the slope of the edge in the x-t plane - higher the velocity, larger the slope., Thus, motion ***detection*** involves detecting an edge in space-time and motion ***measurement*** involves measuring the slope of this edge. Figure 2b shows, a simplistic (and perhaps non-realizable) approach to detecting a spatiotemporally oriented edge and (simultaneously) determining its slope. It essentially requires a ***spatiotemporally oriented*** filter, with positive and negative weighting factors as shown in the figure[1].

With the foregoing discussion on motion perception as edge detection in space-time, we have set the stage for looking at the connection between motion as viewed in space-time and as viewed in spatiotemporal frequency domain. As depicted in Figure 2b, what's needed to locate an oriented edge in space-time is a spatiotemporal filter with its impulse response oriented in space-time. Every filter has a distinct frequency response and so does this hypothetical filter. A typical frequency response of such a filter is shown in Figure 3. It is evident that that the filter is tuned to a band of spatial and temporal frequencies. This can now be seen in light of a 2-d counterpart[2]of Eqn. 1. If we have an infinity of filters, each tuned 'o distinct combination of ω_x and ω_t, only one of these filters will give "significant" response when stimulated with a vertical sine-wave grating moving in a horizontal direction. Knowing ω_x and ω_t of this filter,

[1] It is noteworthy that in an ideal case, we will need an infinity of these filters, each having a different orientation, because the moving bar can ideally have any velocity, resulting into an edge in x-t plane that can have any arbitrary slope.

the velocity can be trivially computed.

3.3 Back to Equation 1: Ideal versus Practical Approach

In this section we discuss the quantitative approach to using Eqn. 1 for computation of the optic flow for a 2-D translating pattern. We will first show the most ideal setting to extract the optic flow and then develop a practical model. In both the cases, for the purpose of simplicity, we begin the analysis with a case where the *image motion* is in only one spatial direction (x-direction, as in Eqn. 2) and then extend the model to general x-y motion. The assumptions on which the model is based are:

- The sequence of images for which the optic-flow is to be determined contains sufficient texture. This assumption is particularly valid in the real life scenes. This assumption guarantees that the aperture problem does not exist. It will be shown later how to extend the model to work in scenes where not enough texture is available and the aperture problem is inherently present.
- All motion is locally translational. That is, in a small spatial neighborhood in the image, the image motion can be assumed to be pure translation.
- All the analytical treatment to follow applies to *each* point on the image. Thus, the model that will be developed will consider a local neighborhood of each point (pixel) on the image to compute the image velocity at the point under consideration. This, iterated (or done in parallel) over all points on the image gives a dense optic-flow array.

The Ideal Case: Consider a pattern translating in the x-direction. The spatial and temporal frequencies must satisfy the relationship shown in Eqn. 2. This shows that the velocity of translation is equal to the slope of the line passing through the origin and the point (ω_x, ω_t) in spatiotemporal frequency plane. In the Fourier jargon.it is customary to characterize this by saying that *the spatiotemporal energy of the translating pattern is concentrated along a line of slope U_x in the (ω_x, ω_t) plane*. The intent, therefore, is to find the slope of the line in the (ω_x, ω_t) plane along which all the *motion energy* is concentrated. A possible, but impractical approach to doing this is shown in Figure 4a. The mechanism consists of an infinity of band-pass (or rather point-pass) filters arranged along a circle, each filter tuned to a unique combination of ω_x and ω_t. Thus if the incoming sequence of images is convolved with each one of these filters, *only one* of them will have a non zero energy in it, the one that lies on the line with a slope of U_x. This slope is immediately known.

This discussion can easily be extended to a 2-D motion. In this case, Eqn. 1 defines the relationship between the spatial and temporal frequencies of the translating pattern. It is evident again, that the orthogonal components U_x and U_y of the velocity of translation are the *direction cosines* of a *plane* in the (ω_x, ω_y, ω_t) space, that satisfies Eqn. 1. In the Fourier jargon again, the *spatiotemporal energy of the translating pattern is concentrated along a plane* in the (ω_x, ω_y, ω_t) space, with direction cosines of its normal as U_x and U_y. The intent therefore is to locate at least two points on the plane so that the plane is uniquelv known (actually three points on a plane are required to determine the plane uniquely, but in our case, the third point is the origin of the (ω_x, ω_y, ω_t) space, and is known). Again, Figure 4b shows a possible, but impractical way to identify the energy plane. This consists of an infinity of frequency selective filters, each tuned to a unique combination of ω_x, ω_y and ω_t, placed all over an infinite cylinder centered around the origin and coaxial with the ω_t axis. If the incoming sequence of images is convolved with each one of these filters, only two of them will have a non-zero motion energy content. Since they lie on the *motion-plane* the direction cosines of the normal to the plane are immediately known and, thus the velocity at the point of translation at the point under consideration is known.

The Practical Case: The foregoing approach is obviously impractical for two reasons.

- It is impossible to design filters tuned a "point" in the 3-frequency space. As we will see in the next section, it is, however, possible to design filters selective to a narrow band of frequencies. They have maximum response at the *center-frequency* of the filter that falls of in a known fashion, depending on the specifications of the filter as the frequency deviates from the center frequency. A typical example of the frequency response of such a filter is shown in Figure 3.
- It is not possible to have an infinite number of frequency selective filters. A practical approach is to have a small number of filters sensitive to the oriented bands of spatiotemporal frequencies and based on the relative energy contained in each of these bands, find an approximate slope of the motion-line (or the direction cosines of the normal to the motion plane). This is explained below for a 1-D motion case.

The practical approach suggested in the foregoing discussion has been reallized in Figure 5a. A similar, but not identical approach was suggested by Heeger [Hee87b]. This consists of five filters placed along a line parallel to the ω_t axis. The power-spectrum of each of these filters is a Gaussian *standing at* the center frequency of the filter in the (ω_x,ω_t) plane. These Gaussians are represented by concentric circles of diminishing radial density as one moves farther away from the center frequency. Evidently each one of these filters will give a non-zero energy when convolved with the incoming sequence of images. The filter whose center frequency is closest to the motion-line will give maximum response and the one farthest will give minimum response.

The energy outputs can be plotted against the temporal frequency (the spatial frequency being constant) as shown in Figure 5b. If a smooth curve is drawn to interpolate (or approximate, under some minimum error criterion) the five known points, the maximum of the curve will correspond to a filter whose center

[2] This relates ω_x and ω_t. The 3-D formulation when one temporal and two spatial dimensions are considered is slightly different and we will discuss it later.

frequency will lie exactly on the motion line. Thus the velocity of translation is known at the point under consideration. Having investigated the principle underlying the extraction of optic flow from a finite number of spatiotemporally oriented filters, we have set the stage for describing the ***pyramid of oriented edges*** that provides a simple scheme for getting a family of ***spatiotemporally oriented filters***, the basic hardware for optic flow extraction.

3.4 The Pyramid of Oriented Edges: A Family of Spatiotemporally Oriented Filters

It was clearly brought out in the previous paragraphs that filters with spatiotemorally oriented impulse response are sensitive to motion information in an image sequence. Henceforth, in the ensuing discussion, these spatiotemporally oriented filters will be referred to as motion sensitive filters also, for obvious reasons. What is needed to *detect* and *measure* motion is a *set of motion sensitive filters* strategically located in the 3-frequency plane and a *set of procedures* to extract optic-flow from the relative magnitude of the energy content of these filters, respectively. Over the past few years, researchers in physiology as well as computational vision have proposed a variety of motion sensitive filters. With a very few exceptions, most of the reported research [WA83], [FJ85], [AA85], [AB85] [MU81] describes only how to construct motion sensitive filters, without a detailed treatment of how to extract optic-flow, given the motion sensitive filters. We describe below a new image structure, that we call a *Pyramid of Oriented Edges* or a POE, that provides a family of filters, each member of which occupies a band of spatiotemporal frequencies that *is oriented in the spatiotemporal-frequency space.* Seen in the space-time domain, each member of this family of filters is sensitive to an edge with a specific orientation in space-time and hence is *motion-sensitive*.

To keep the discussion simple, we will first describe a 2-D version of the POE. This version is is essentially a *spatial* pyramid with *no temporal dimension*. This version of the model works on a single image and decomposes it into several "channels", each channel sensitive to spatial edge features in a specific direction. Functionally, therefore, each channel is an orientation selective filter in *space only*. We will discuss the issues involved in an extension of the 2-D POE to incorporate the temporal dimension in the end. The 3-D POE thus obtained will have channels selective to oriented edges in space time and hence, will be motion sensitive.

We explain the computational approach taken in the model using a specific instance that has three channels at each level of the pyramid, sensitive to edge features in horizontal, vertical and diagonal directions as shown in Figure 6. The computational scheme to construct the POE is a very logical extension of Burt's pyramid representations. A brief review of Burt's representation and its proposed extension is given below in a qualitative sense. Burt's representations consists of two pyramids, namely Gaussian pyramid and Laplacian pyramid. Gaussian pyramid contains low-pass filtered copies of the original image, at successively decreasing resolutions. Laplacian pyramid, on the other hand, contains band-pass filtered copies of the original image. A representative example is shown in Figure 7a (after Burt).

The computations involved in constructing, say, L_i and G_{i+1}, given G_i (the original image being G_0), can be visualized easily from Figure 7b. In this figure, H and V refer to Gaussian filter in the horizontal and vertical direction, as defined by Burt. The characteristic property of these filters that is of relevance here is that the spatial frequency content of the filtered image is half of that of the input image. Thus, for example, if an image is convolved with H, the frequency content of the filtered image, in the *horizontal* direction, is reduced to half its original value. This is depicted in Figure 7b(i). Similarly, Figure 7b(ii) shows an image (G_i) filtered with H and V successively. Since the overall frequency content of the resultant image is half the original value it is justified to *decimate* the resultant image by sampling every other pixel, both along rows and columns, without introducing any aliasing. This decimated image is G_{i+1}. This decimated image can be expanded to twice its current size by an EXPAND operation described later in this section. If we subtract the EXPANDed version of G_{i+1} from G_i, what happens in terms of the frequency content is displayed in Figure 7b(iii). It is easy to see that the resultant image is nothing but L_i, a band-pass-filtered image depicting the edges.

A qualitative description of the extension proposed for orientation selectivity can be easily visualized from Figure 8. Basically L_i is convolved with H, V and (1-H-V) respectively. The results are shown in Figure 8b, 8c and 8d respectively. It is apparent that these three images depict the spatial edge features oriented in horizontal, vertical and diagonal directions respectively. *These three images, thus comprise the Level$_i$ of the POE.* The three images at the *Level$_i$* of the POE, which we denote by LH_i, LV_i and LD_i (standing for images depicting horizontal, vertical and diagonal intensity gradients respectively), can be quantitatively described as

$$LH_i = L_i * H$$

$$LV_i = L_i * V$$

$$LD_i = L_i - L_i * H - L_i * V$$

The extension of this modelto a general scheme with K channels instead of the three described here is not included here for purpose of brevity. The interested reader is refered to [SR87] for details.

However, the extension of the 2-D POE to three dimensions, to *include the temporal processing* is worth mentioning. We have implemented this scheme in a *space-time separable fashion*. Each image of the image-sequence undergoes spatial band-pass filtering to give corresponding image sequences that depict oriented edge features. Each one of these sequences is convolved with a difference of *temporal* Gaussians to get the temporal band-pass effect. the net effect of these two steps of processing is a family of spatiotemporal band-pass filters that are sensitive to oriented edges in space-time, or alternatively, that are *motion sensitive*. This is schematically depicted in Figure 9. This seemingly straightforward extension of the POE to include the temporal dimension is computationally very demanding. Specifically, to do it in real time also calls for a very careful synchronization in addition to the increased number crunching. This aspect will be discussed later along with other issues of real time implementation.

4 Doing it in Real Time

Most of the practical applications of a computational model for motion perception, such as robotics and object tracking in space applications etc., inherently call for a *real time* implementation of the model. This aspect has largely been ignored in thepast research. We have investigated the underlying issues and implemented our model in real time on the PIPE system. It is apparent from the foregoing discussion that the nature of computations involved in the models described above is very local in both space and time. It is therefore worthwhile reinterpreting the basic definitions to develop their *pixel-parallel* versions that can then be mapped on a variety of state of the art image processing architectures that support (i) a parallel execution of simple operations on all the pixels in the image and (ii) concurrent temporal processing.

We first briefly review one such architecture, namely PIPE on which we have implemented the abovementioned models in real time. We then briefly discuss the approach to developing the "pixel-parallel" version of Burt's pyramid and its extension to the POE and contrast it with that of Mallat's pyramid[3]. For a quick reference, the algorithm to construct Mallat's pyramid is shown pictorially in Figure 10. The interested reader is referred to Mallat's original work for details.

More importantly, we show how to include temporal processing in the POE in the pixel-parallel versions using the space-time separable strategy described before. The pixel-parallel versions map directly onto the PIPE architecture. We then show the results of the PIPE implementation. *The results are also available on a video-tape to display the real-time nature of the results.*

4.1 A Review of the PIPE Architecture

PIPE is a multi-stage parallel processor designed to process images at video-rate. Each stage in the sỹstem, called a *Modular processing Stage* (MPS), is designed so that all input, processing and output are completely synchronous with the video-raster and thus allows complete image to be treated as one data-structure. A schematic diagram of the details of the architectures is shown in Figure 9. Figure 9a shows the connectivity of the eight stages. There is a *forward path* connecting the output of *each* stage to the input of the next stage, a *backward path* connecting the output of *each* stage to the input of the previous stage and a *recursive path* connecting the output of *each* stage to its own input. In addition, there are six video buses to connect the output of any stage to the input of any other. Each of these data-paths is eight-bits wide.. Images can be made to stream between stages, spending one cycle (1/60th of a second) in each stage for processing. Figure 9b shows the capability available within each stage. The hardware modules available in each stage include (i) two *image buffers*, 256x256x8 bits each, for storing images, (ii) two *neighborhood operators* to do any arbitrary 3x3 or 9x1 convolution on the complete image, (iii) four *look-up table operators* to do any arbitrary point transformation operation on the complete image, such as multiplying each pixel in the image by 2, (iv) three ALU's to do simple operations on two images, such as subtracting one image from the other, pixel by pixel and (v) a *Two Valued Function Module (TVF)*, that is a very powerful tool to do any arbitrary operation on two images, or to perform arbitrary image warping operation operations, such as rotating an image by an arbitrary angle. It needs to be emphasized that all the operations in a stage can be done on the *complete* image, and can be finished in *1/60th of a second*.

4.2 Pixel Parallel Versions of the Proposed Models

The key issue in developing pixel-parallel versions is to exploit the spatial and temporal parallelism inherent in the computation model. For purpose of brevity, we will not describe the complete algorithms here. Rather, we summarize the salient points here. The interested reader is referred to [Sin87a].

The spatial operations used in the model described above comprise convolutions over small spatial neighborhoods, a fixed unary operation on all pixels of an image, a fixed binary operation on all the corresponding pixels of two images, subsampling to reduce image size or pixel replication to expand image size etc. They point to an SIMD approach to spatial parallelism that is supported by the PIPE, as described before. The temporal filtering required to make the POE channels motion sensitive was described earlier. That is supported very well by the PIPE's interstage connectivity. The forward, backward and recursive paths allow the spatial processing to be carried out in three successive stages and then the three consecutive spatially filtered images can be combined via the three paths described above. This concurrency is one of the key aspects of real time implementation.

4.3 Implementation Results

The algorithms described above were implemented on the Eight-Stage PIPE. The results of the implementation are discussed below.

For Burt's pyramid and the POE, the current implementation computes three levels of the Gaussian pyramid, two levels of the Laplacian pyramid and one level of the 2-D POE. quite acceptable. It takes four cycles of the PIPE to do all these computations. So a new image is sampled every (1/15)th of a second, i.e., at one fourth of the standard video rate and the abovementioned levels of the pyramid for an image are available, just before the next image is ready to be sampled. The results are shown in Figure 11a.

[3] It is noteworthy that Mallat's pyramid [Mal87] is functionally identical to the *spatial* POE with three channels described above. It however uses *quadrature mirror filter* pairs, or QMF pairs that are computationally much more expansive than the simple Gaussian filter used in the POE. QMF pairs, however provide very good "tuning" and hence their 3-D extensions to include the temporal dimension have potential application to optic-flow computation. Heeger [Hee87a] considers them a possible replacement for the Gabor filters he used in his model. We are currently investigating the issue of extending Mallat's pyramid to three dimensions, in fashion similar to the 3-D POE, whose implementation has been discussed above.

We have also implemented separately, one level of the 3-D spatiotemporal POE following the strategy of Figure 9. This computes optic-flow by generating *three* image sequences sensitive to image motion in horizontal, vertical and diagonal directions respectively. One frame from the incoming image sequence, and one frame from the output sequence, depicting the image velocity is shown in Figure 11c. The scene comprised of the UTAH-MIT dextrous hand in our laboratory. Only two of the three fingers visible in the input image (top and bottom fingers) were set in motion. The middle finger was kept stationary. This is apparent in the output image. Where the contours of the top and the bottom fingers depict motion. This takes three PIPE cycles, and thus a new optic-flow is available every 1/20th of a second. These speeds show promise for real-time object tracking in robotics. We are currently working on increasing the number of direction channels in optic flow computation beyond three in the current version.

For Mallat's pyramid we have implemented a version that constructs one level of the pyramid. We used a 9x1 low-pass and band-pass QMF. Since a 9x1 convolution is currently not available on the PIPE, it was implemented using the 3x3 masks by the parallel technique described by [Sin87b]. This consumes a lot of PIPE cycles, and hence, the current version uses 20 cycles. Thus a new image is sampled every one third of a second and the results are available just before the next image is ready to be sampled. However it has been calculated that with the 9x1 convolution mask available in hardware, the same computation can be done in 6 PIPE cycles, i.e., a new result is available every one tenth of a second. Figure 11b shows the results for one level of Mallat's pyramid.

All these results are available on a video-tape to show the real time nature

5 Conclusion

In this paper we have made a strong case for hierarchical image representations and processing as an effective tool for robust optic-flow measurements. We have identified two classes of computational models in which they can be applied to compute optic-flow from a sequence of images. After examining this observation in light of the past research, we have described a new hierarchical image structure, the *pyramid of oriented edges*, that can be used to generate a family of motion sensitive filters. Motion sensitive filters, or spatiotemporal orientation selective filters have been identified in the recent research as the basic hardware for optic-flow computation based on spatiotemporal frequency approach. We have also emphasized the need to implement the basic structures required for optic-flow computation in real time. For this purpose, we have developed pixel-parallel versions of two hierarchical schemes, namely our own POE and the functionally similar Mallat's pyramid and compare their time-performance via real-time implementation on the PIPE architecture. We have implemented the 3-D version of the POE to compute a coarse form of optic-flow. As a prerequisite to the POE, we have also implemented the pixel-parallel versions of Burt's pyramid. It is noteworthy that this implementation is useful not only for our spatiotemporal frequency based model, but can also serve as a "core" for the real-time implementation of optic-flow computation schemes based on intensity-correlation based methods such as that of Anandan.

This work is a part of our ongoing effort to get very robust and fast measurements of optic-flow. Our plans for the near future include (i) developing a scheme on the lines of Figure 5b to extract the true optic flow from the motion sensitive filters obtained with the POE, (ii) implement two different optic-flow schemes, namely an intensity correlation based model similar to that of Anandan and our own POE based model, around the real-time Burt's pyramid and the POE respectively and finally (iii) integrate the above two models to get a robust optic-flow scheme. Intensity correlation methods tend to work very well in "structured scenes" whereas spatiotemporal frequency based methods are more suited for textured scenes. Their integration can, therefore, be a potential source of robustness over a wide range of scenes.

Acknowledgement: The authors would like to thank Surendra Ranganath of Philips Laboratories, Briarcliff Manor for his useful discussions related to this work.

References

[AA85] Watson. A.B. and A.J. Ahumada. Model of human visual motion sensing. *J. Opt. Soc. Am.*, 2 No. 2:322–341, 1985.

[AB85] E.H. Adelson and J.R. Bergen. Spatiotemporal energy models for the perception of motion. *Journal of the Optical Society of America*, 2(2):284–299, 1985.

[Ana86a] P. Anandan. *Computing Optical Flow from Two Frames of an Image Sequence.* Technical Report COINS-TR-86-16, COINS Department, University of Massachusetts, Amherst, 1986.

[Ana86b] P. Anandan. *Computing Optical Flow from Two Frames of an Image Sequence.* Technical Report COINS-TR-86-16, COINS Department, University of Massachusetts, Amherst, 1986.

[Ana86c] P. Anandan. *A Review of Stereopsis and Motion.* Technical Report COINS-TR-86-18, COINS Department, University of Massachusetts, Amherst, 1986.

[Bur84] P.J. Burt. The pyramid as a structure for efficient computation. In A. Rosenfeld, editor, *Multi Resolution Image Processing and Analysis*, pages 6–37, Springer Verlag, 1984.

[Cro84] J.L. Crowley. A multi resolution representation for shape. In A. Rosenfeld, editor, *Multi Resolution Image Processing and Analysis*, pages 169–189, Springer Verlag, 1984.

[FJ85] D.J. Fleet and A.D. Jepson. *On Hierarchical Construction of Orientation and Velocity Selective Filters.* Technical Report RBCV-TR-85-8, Department of Computer Science, University of Toronto, 1985.

[Gla87] F.C. Glazer. *Hierarchical Motion Detection*. PhD thesis, Department of Computer and Information Science, University of Massachusetts, Amherst, COINS-TR-87-02, 1987.

[Hee87a] D. Heeger. In *Personal Communication*, 1987.

[Hee87b] D. Heeger. A model for extraction of image flow. In *First International Conference on Computer Vision*, 1987.

[HS81] B.K.P Horn and B. Schunk. Determining optical flow. *Artificial Intelligence*, 17:185–203, 1981.

[Mal87] S.G. Mallat. *A Theory for Multi Resolution Signal Decomposition: The Wavelet Representation*. Technical Report MS-CIS-87-22, Department of Computer and Information Science, University of Pennsylvania, 1987.

[MU81] D. Marr and S. Ullman. The role of directional selectivity in early visual processing. *Proceedings of the Royal Society of London*, B-211:151–180, 1981.

[Shn84] M. Shneier. Multiresolution feature encodings. In A. Rosenfeld, editor, *Multi Resolution Image Processing and Analysis*, pages 190–199, Springer Verlag, 1984.

[Sin87a] A. Singh. *Constructing Multiresolution Image Pyramids in Real Time*. Technical Report, Department of Computer Science, Columbia University, CUCS-TR-286-87, 1987.

[Sin87b] A. Singh. *Image Processing on PIPE*. Technical Report TN-87-093, Philips Laboratories, Briarcliff Manor, New York, 1987.

[Sin87c] A. Singh. *Measurement of Visual Motion*. Technical Report, Department of Computer Science, Columbia University, CUCS-TR-287-87, 1987.

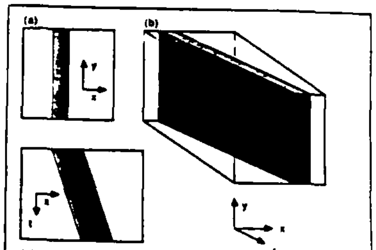
[SR87] A. Singh and S. Ranganath. A heirarchical model for directional selectivity of visual motion. *In Preparation*, 1987.

[TB87] W.B. Thompson and S.T. Barnard. Lower level estimation and interpretation of visual motion. *IEEE Computer*, August:20–28, 1987.

[The87] *The PIPE User's Manual*. Aspex, 530 Broadway, New York, NY 10012, 1987.

[WA83] A.B. Watson and A.J. Ahumada. *A Look at Motion in Frequency Domain*. Technical Report NASA Technical Memorandum 84352, NASA Ames Research Center, Moffett Field, CA 94035, 1983.

[Wax84] A.M. Waxman. An image flow paradigm. In *Proceedings of the 2nd IEEE Workshop on Computer Vision, Annapolis*, pages 49–57, 1984.



(c) Figure 1 A Vertical Bar Moving to Right
(a) One Snapshot
(b) Spatiotemporal Cuboid
(c) An x-t Slice from Spatiotemporal Cuboid
(After Adelson and Bergen)

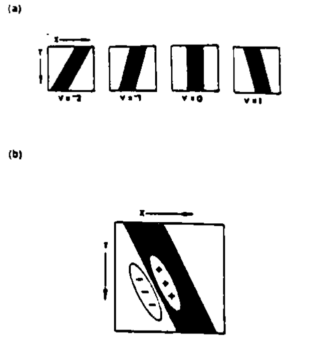


Figure 2: Motion as Orientation in Space Time
(a) The x-t Slice for Different Velocities
(b) A Spatiotemporally Oriented Filter to Detect Motion
(After Adelson and Bergen)

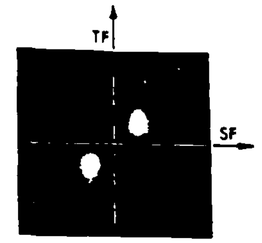


Figure 3: The Spatiotemporal Energy Spectrum of an Orientation Selective Filter
(After Adelson and Bergen)

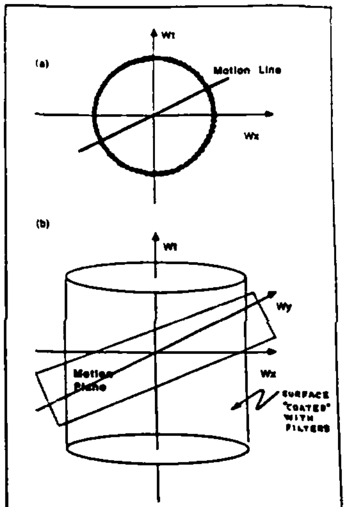


Figure 4: Extracting Velocity from Motion Sensitive Filters.
(a). 1-D Motion: Infinity of Filters Along a Circle
(b). 2-D Motion: Infinity of Filters Along a Cylinder

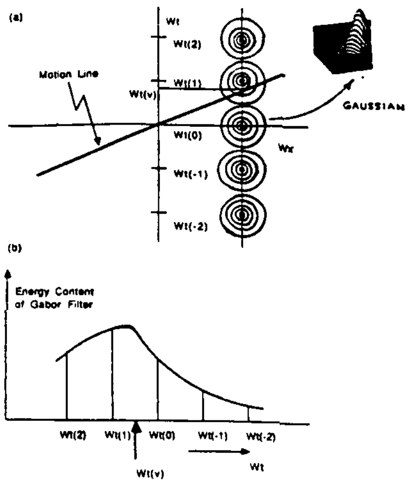


Figure 5: A Practical Approach to Velocity Determination
(a). A Strategic Placement of Five Gabor Filters in Wx-Wt Plane
(b): The Temporal Frequency Wt(v) Corresponding to the Actual Velocity is at the MAX of the Interpolating Function

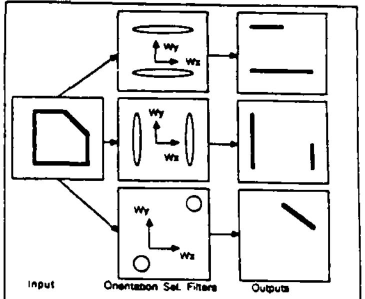


Figure 6 : A Set of Filters Selective to Horizontal, Vertical and Diagonal Orientations

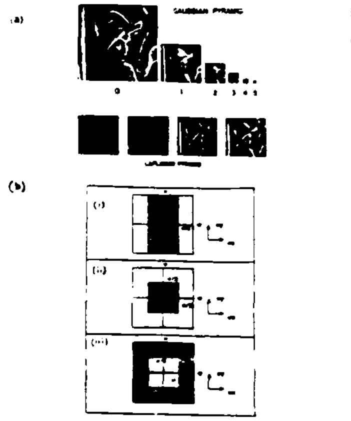
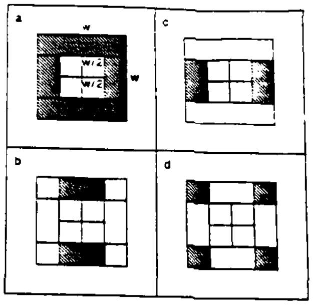


Figure 7 . Burt's Pyramid
(a). An Example of Gaussian and Laplacian Pyramid
(b): Filtering Operations For Constructing the Pyramid



a) The Frequency Content of L(i)
b) The Frequency Content of L(i) * H
c) The Frequency Content of L(i) * V
d) The Frequency Content of L(i) * (1-H-V)

Figure 8 Constructing Orientation Selective Filters By ·ectional Smoothing of the Laplacian Filter

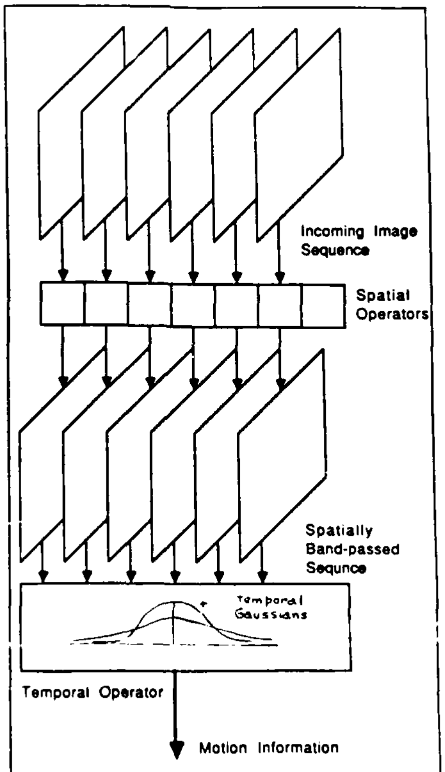


Figure 9: A Schematic Diagram For Space-Time Separable Filtering

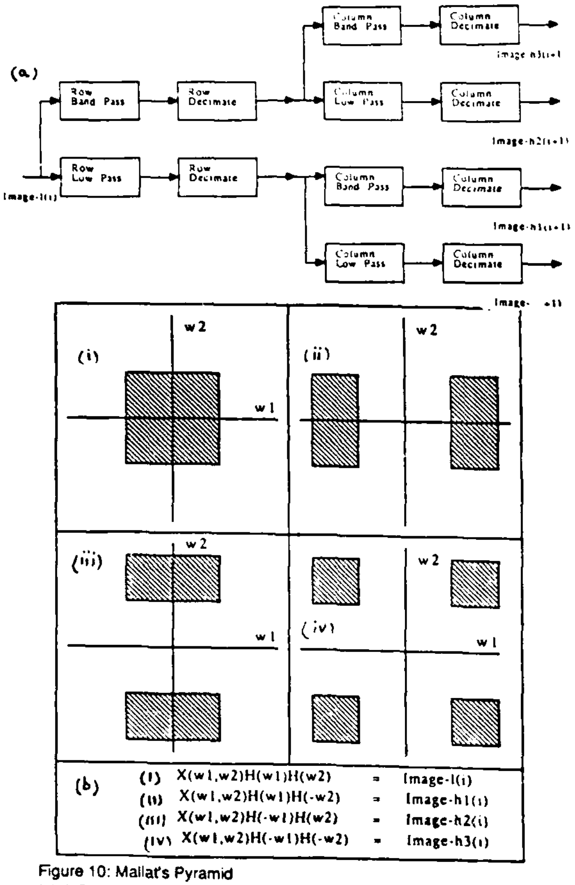
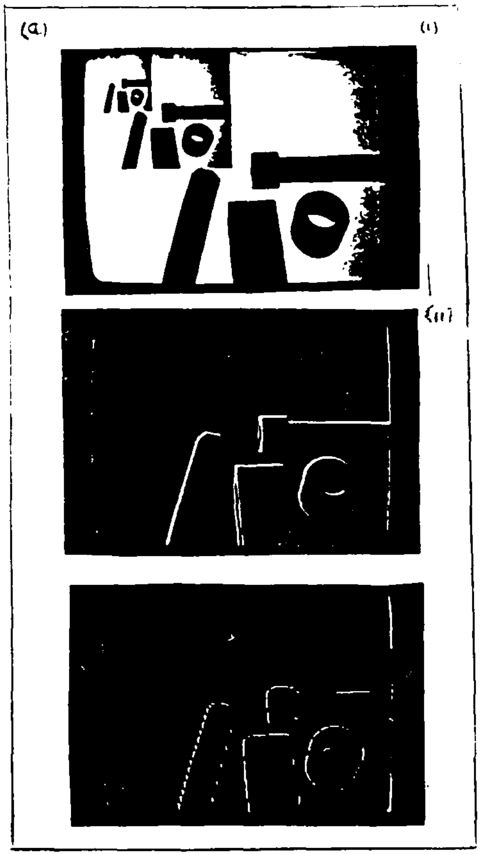
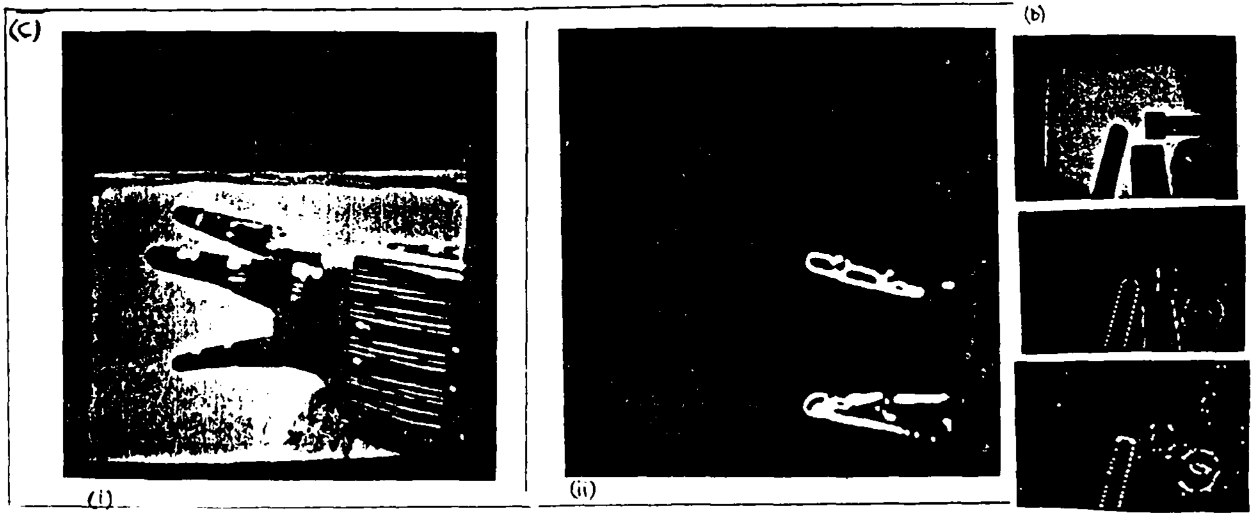


Figure 10: Mallat's Pyramid
(a) A Schematic Representation of the Algorithm (After Mailat)
(b) The Frequency Content of Various Components in (a)
(Note: i and i+1 denote two consecutive levels of the pyram

Figure 11: Implementation Results
a) (i) Three Levels of Burt's Gaussian Pyramid
(ii) Two Levels of Burt's Laplacian Pyramid (Expanded to Full Size
b) Mallat's Pyramid - Image-I(1), Image-h2(1) and Image-h3(1) at the level-1 (bottom level).
c) (i) One Frame from the Incoming Image Sequence for construction of the POE (Only *Top* and *Bottom Fingers* of the UTAH-MIT dextrous hand seen here were set into motion)
(ii) Image Velocity
(*Note that the motion is depicted only in the contours of the two fingers in motion*).





TRANSLATING OPTICAL FLOW INTO TOKEN MATCHES

Lance R. Williams and Allen R. Hanson

Department of Computer and Information Scieuce
University of Massachusetts
Amherst, Massachusetts

ABSTRACT

Motion in the environment manifests itself in changes of many kinds, not just image plane velocities, yet these are all that an optical flow field makes explicit. We view optical flow as a useful low-level representation from which a symbolic description of change, in the form of token matches, can be computed. The tokens of interest to us are those produced by perceptual organization processes, and are more abstract than edges or interest points. We demonstrate a working system for matching line tokens which uses the optical flow field in a heuristic manner to limit the search for the *minimal bipartite cover* of the set of tokens from each frame.

1. INTRODUCTION

It is our position that the inherently local measurement of visual motion provided by optical flow is insufficient to meet the varied requirements of dynamic image understanding. We choose to describe the time varying image by computing correspondence between tokens of arbitrary spatial scale produced by perceptual organization processes. We believe that this will result not only in more accurate measurement of visual motion, but also facilitate the use of motion information in object recognition and scene understanding.

For the purposes of this paper, techniques for measuring visual motion can be roughly categorized as either optical flow methods or token matching methods. This taxonomy is based upon the nature of the output representation. Specifically, a distinction is drawn between methods whose purpose is the computation of a velocity or displacement field, and methods which compute correspondence in time between tokens that serve as descriptors of spatial structure.

The goal of the optical flow methods is to compute a vector function of the image plane. Depending on the particular method, each vector represents either a velocity or a displacement. The typically pixel-parallel nature of the computation is dictated largely by the form of the input and output representations, which are arrays of values in registration with the original scene. Optical flow methods have been more successful than token matching methods, and much of this success is due to the fact that these methods implicitly exploit information carried directly in the "shape" of the image intensity surface. This is usually effected through the use of an *intensity constancy constraint* (e.g. See Horn and Schunk [8]). The utility of optical flow methods has been further increased by the ease with which they can be expressed hierarchically, resulting in faster algorithms capable of describing larger motions [1,13,5].

The token matching methods which concern us compute correspondence in time between spatial structures produced by grouping processes. These tokens belong to what Marr has called the *full primal sketch*, [10] and are more abstract than edge segments [11] or interest points [2,12,15]. Tokens map directly to environmental structure, and descriptions of their movement correlate more closely with the motion of physical objects, than does optical flow. Most importantly, token matching allows change through time to be expressed in a wide variety of ways. A token match represents more than a spatial displacement, also explicit are the changing values of any parameters associated with the token. These can include orientation, length, area, contrast, color, etc. In short, motion in the environment manifests itself in changes of many kinds, not just as image plane velocities. Information of this kind can facilitate interpretation, including computation of environmental depth (see [18] in these proceedings). While this information is explicit in the output of a token matching method, it is difficult or impossible to compute from an optical flow field.

While spatial structure is better described by a set of tokens than by an array of values, current perceptual organization processes fail to adequately describe the shape of the image intensity surface. Indeed, this information is usually intentionally discarded during the abstraction process. However, even if a sufficiently powerful descriptive language were developed (.e.g. [3,6]), and a token matching approach were formulated, it is difficult to see how such an approach could rival the efficiency and simplicity of methods which exploit this information implicitly. through an intensity constancy constraint.

The fact that a representation is easy to compute reveals nothing about its utility. Interpretation of an optical flow field independent of any *knowledge of the spatial structure from which* it was derived, seems difficult at best. Spatial structure can be characterized locally with an interest operator [12] and interpretation can be restricted to the sparse set of points with a high interest operator score. Alternatively, local structural measures can be incorporated within the optical flow computation itself, allowing a dense flow field to be computed through a *smoothness constraint*. Nagel employs a second order approximation of the intensity variation to determine the direction in which a smoothness constraint is enforced [14]. Anandan analyzes the

[1] This research is sponsored by the Rome Air Development Center (RADC) and the Defense Advanced Research Projects Agency (DARPA) under contract F30602-87-C-0140, and the Defense Mapping Agency (DMA) under contract DMA800-85-C-0012.

principle curvatures of the sum-of-squared-difference surface to associate vector confidence measures with each displacement, which in turn, indirectly influence the enforcement of a smoothness constraint [1]. The characterizations of spatial structure in the techniques of Anandan and Nagel are simple and local, as is necessitated by the need to incorporate such characterizations within the formulations of the smoothness constraints.

Recent work in perceptual organization has given us a richer vocabulary with which to describe spatial structure than has been available in the past [9,10,16,19]. While local operators are useful for detecting local structures, more powerful grouping processes are required to recognize structure of arbitrary scale. Since the image structure revealed through grouping corresponds directly with environmental structure [19], perceptual organization provides the best measure of "interest." Recently, Boldt has incorporated these ideas in a working program for grouping long straight line segments [4,17].

With this in mind, a logical course of action might be to enforce a smoothness constraint along the length of a line segment produced by a perceptual organization process such as Boldt's. In fact, Hildreth's smoothness formulation can be enforced along an arbitrary contour, and for the specific case of a line segment in three space undergoing rigid motion, she demonstrates that it yields the physically correct flow [7]. If our goal were to produce a better flow field, this is exactly what we would do, but it has already been suggested that knowledge of correspondence in time between the line segments themselves would result in a representation more useful to the interpretation task. We choose to view the optical flow field as a useful low-level representation from which a symbolic description of change, in the form of token matches, can be computed.

2. GROUPING FAILURE

As tokens become more abstract, they also become more unique. Therefore, it is often assumed that the more abstract a token is, the less ambiguous matching will be. To a certain extent, this is true, but increased abstraction brings with it a new source of ambiguity. Under ideal conditions, we might expect two perceptual organization processes operating independently on two frames of a motion sequence to partition each frame into the same set of tokens. After all, each frame is a slightly different view of a predominantly stable physical world separated only slightly in time. Unfortunately, in practice, due to the discrete sampling of image formation (and other image effects such as shadows, highlights, and texture), the likelihood of two frames being partitioned into the same set of tokens is very small. Since the basic operation employed in perceptual organization is grouping, discrepencies can arise in two ways: 1) failure to relate two tokens that have a single physical cause, or *undergrouping*, and 2) mistakingly relating two tokens that have separate physical causes, or *overgrouping*. While both undergrouping and overgrouping can be caused by noise, overgrouping errors most often occur when two tokens satisfy the geometric criteria for grouping through pure chance. Unfortunately, this happens more frequently in motion sequences depicting the view of the world from the vantage point of a moving sensor. While the odds of two unrelated tokens accidently satisfying the grouping criteria in any single view are small, the odds of the moving sensor passing through such degenerate views in the course of a motion sequence are much higher. The solution to this problem is beyond the scope of a simple, two-frame matching approach, and it will be discussed in greater detail when suggestions for a multiple frame approach are presented later in the paper.

Even relatively simple undergrouping errors, rule out the possibility of a one-to-one mapping between tokens from successive frames, since the number of tokens in each frame will rarely be the same. Under these circumstances, it seems that the best mapping possible is a mapping that assigns each token at least one match, and optimizes some error function in the process. Such a mapping is called a *minimal bipartite cover*. We first encountered the minimal bipartite cover, in the context of the correspondence problem, as part of Ullman's *minimal mapping theory* [20]. Interestingly, the motivation Ullman gives for its use is unrelated to the grouping failure argument presented here. We believe that the minimal bipartite cover is simply a more practical goal than a one-to-one mapping when matching abstract tokens prone to grouping errors.

3. FROM OPTICAL FLOW TO TOKEN MATCHES

Ullman's minimal mapping theory, which presents the correspondence problem as an optimization problem in the abstract, is a very general paradigm, and it serves as a useful point of departure for the discussion of the method proposed in this paper.

In the minimal mapping theory, correspondence is computed between tokens from two frames by finding the minimal bipartite cover of the graph whose nodes are the tokens from each frame and whose arcs reflect potential correspondence. The weight of each arc in the graph is called an *affinity* measure, and is a function of the relative similiarity and spatial separation of the two tokens which the arc links. Ullman justifies his choice of particular affinity values with data from studies of the human visual system. Indeed, the minimal mapping theory is offered as a possible explanation for the manner in which many of the classic Gestalt displays such as Ternus' configuration are interpreted by the human visual system. Because of the explosively large number of possible mappings for matching problems of even modest size, Ullman simplifies the general matching problem by assuming that the number of candidate matches that each token can claim is equal to some small integer constant. He then shows, that under this assumption, the optimization problem can be solved by a hill climbing process, which leads to a relaxation algorithm. However, no method for choosing initial candidate matches is offered, and the number of iterations required for convergence is unclear.

The approach described in this paper reflects a natural synthesis of the optical flow and token matching paradigms. Because the optical flow field is a vector function of the image plane, it can be used to define a transformation that maps tokens from one frame to their predicted positions in the next frame. The spatial area that must be searched for a potential match is reduced to a small region surrounding the token's predicted position. Furthermore, the merit of a potential match is judged by its proximity to its predicted position, not to its previous position. This simplifies the general matching problem by restricting the number of candidate matches.

4. AN IMPLEMENTATION USING LINE TOKENS

In this section, we describe an implementation of the general approach just outlined. A schematic view of the implementation is depicted in Figure 1.

In a preliminary step, a set of line tokens is computed for each frame using Boldt's line grouping algorithm (Figures 2 and 3). Boldt's algorithm begins by extracting an initial set of line segments whose orientation is the normal to the gradient direction along zero crossing contours of the Laplacian operator. These initial line segments form the nodes of a graph whose arcs

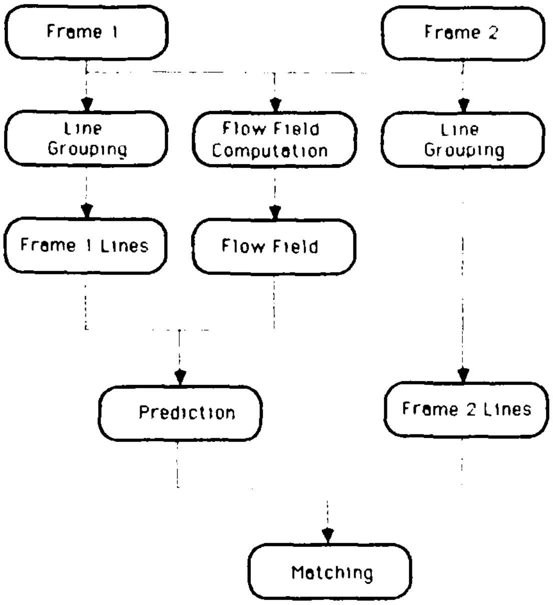


Figure 1. Information flow diagram for an implementation using line tokens.

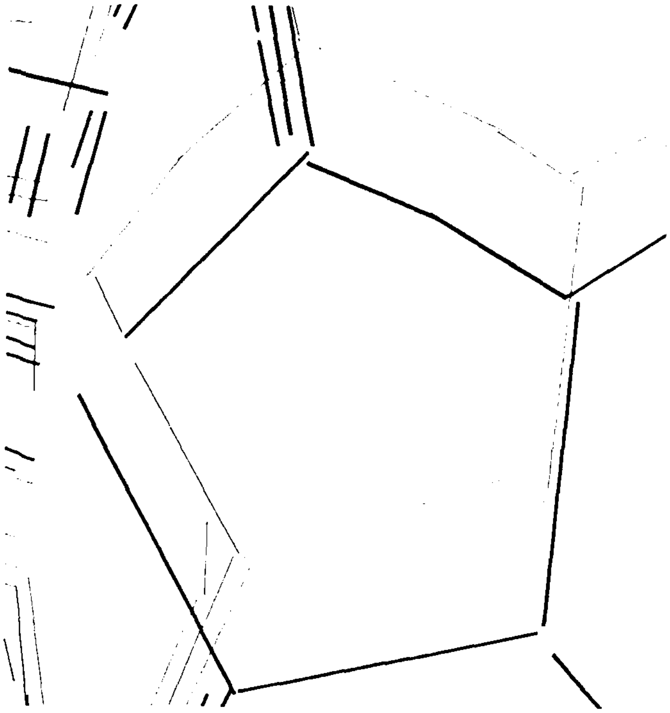


Figure 3. Line tokens computed with Boldt's grouping algorithm. Lines for frame one are displayed thick while lines for frame two are displayed thin.

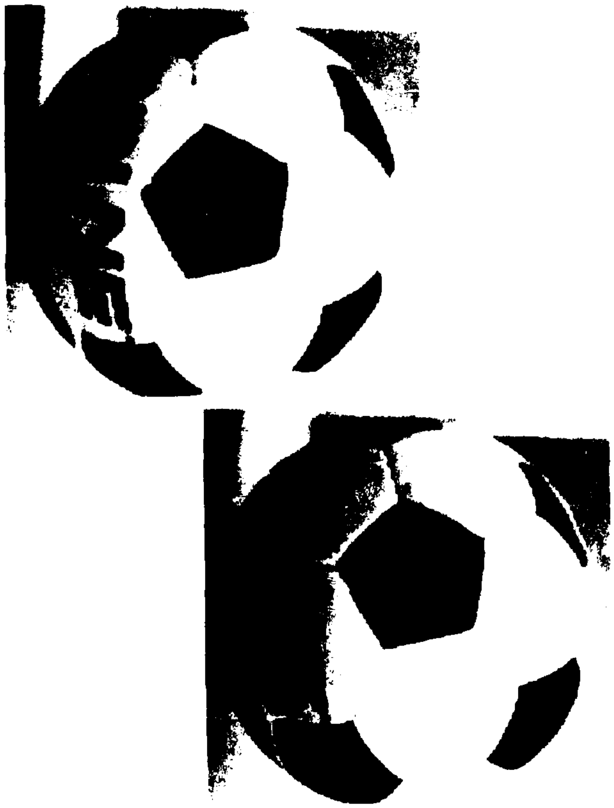


Figure 2. Frame one and two of a sequence depicting a moving soccer ball

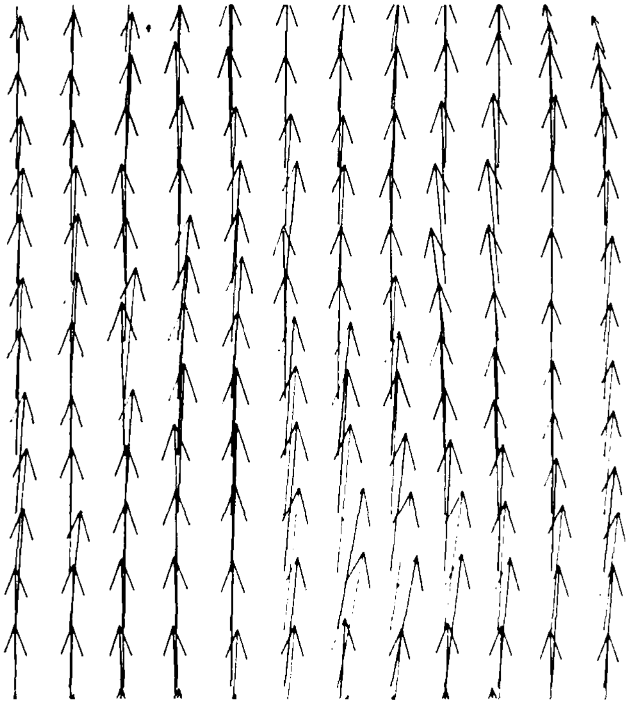


Figure 4. Sample of the displacement vectors computed with Anandan's algorithm.

(*links* in Boldt's terminology) reflect a significant non-accidental geometric relationship between the two line segments they join. Some of the relations used as *linking criteria* are endpoint proximity, orientation difference, lateral distance, overlap and contrast difference. All paths through the *link graph* within the current *replacement radius* are examined and the path minimizing the mean-square-error of a straight line fit is replaced by a new line segment. The program is then invoked recursively on the new set of line segments, using a larger replacement radius, resulting in ever smaller sets of increasingly longer lines. A final set of between one and two hundred lines is produced by filtering on length and contrast.

In a second preliminary step, the optical flow field is computed using the method developed by Anandan [1] (Figure 4). Strictly speaking, Anandan's algorithm produces a displacement field, not a flow field. The intensity constancy constraint exists implicitly as a sum-of-squared-difference measure within a Laplacian pyramid. Anandan's algorithm uses knowledge of the direction of principle curvature of the sum-of-squared-difference surface to enforce a smoothness constraint at each level of the laplacian pyramid. These design choices together comprise a working system that appears to consistently yield reliable estimates of image displacements.

The predicted position for each line from the first frame is computed by a least squares fit to the points comprising the image of that line under the transformation defined by the optical flow field. All lines from the second frame passing through a narrow rectangular region surrounding this predicted position are retrieved (Figure 5). The size of the search region is a parameter of the system. Although currently a constant, it could conceivably be coupled to the value of confidence measures associated with the optical flow, such as those computed by Anandan's algorithm. In this way, the window size would be smaller in areas of high confidence and larger in areas of low confidence.

A bipartite graph, henceforward called the *time-link graph*, is constructed. Its arcs connect line segments from the frame one token set, to all candidate matches retrieved from the second frame (Figure 6). The weight of each arc in the time-link graph is a measure of the discrepancy in position between the predicted position of the frame one line segment and the position of the candidate match. Since the line segments' lengths are highly unstable (because of undergrouping errors) information about length is not incorporated into the positional discrepancy metric. Instead, the measure approximates the average distance between the two segments, independent of their lengths (Figure 7). Ideally, one would use a measure similiar to that employed by Lowe in his model matching system [9], which computes the probability of the juxtaposition of two line segments being due to chance alone, using knowledge of the distribution of background line segments.

By computing the connected-components of the time-link graph, the global matching problem is conveniently divided into smaller, individually tractable pieces which reflect the scope of potential interactions. For each connected-component, the bipartite cover minimizing the positional discrepancy metric is found. This is accomplished through a simple blind search of the sub-graphs of each connected-component. Although Ullman suggests solving the optimization problem through network relaxation, the need for such an approach is eliminated here because of the relatively small size of each connected-component. Indeed, a connected-component often contains only a single arc, in which case the match is uniquely determined. This is directly due to the heuristic use of the optical flow field. The bipartite

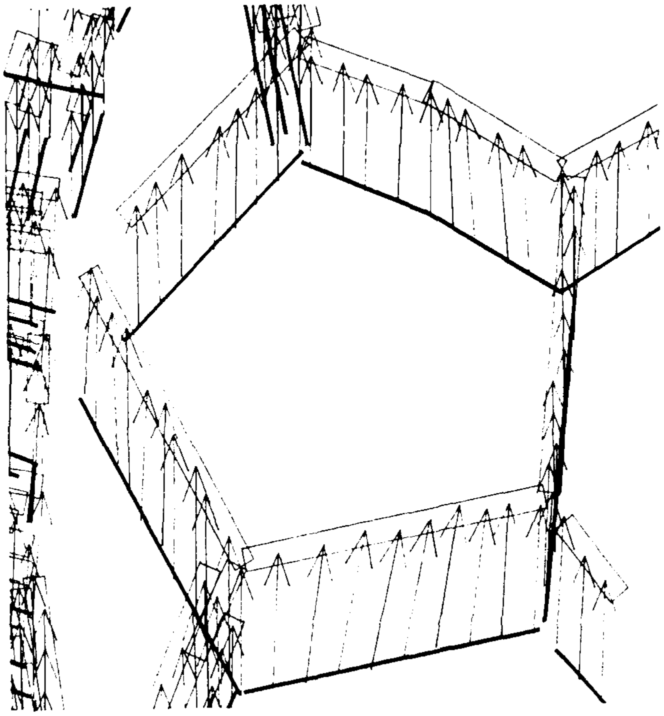


Figure 5. The rectangular search regions computed for each line token by a least squares fit to the tips of the displacement vectors. Lines from frame two that intersect the search region become candidate matches.

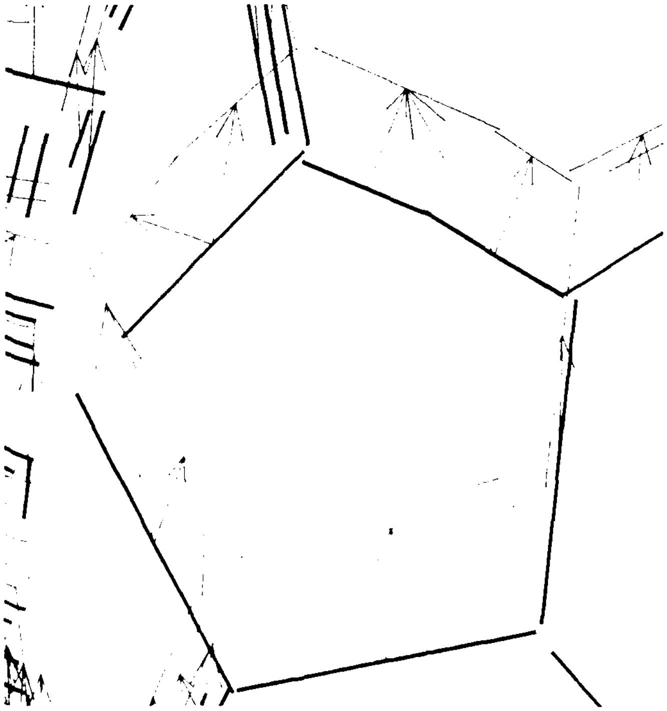


Figure 6. The *time-link graph*, where the arcs connect lines in frame one with their candidate matches.

cover reflects the final correspondences reported by the system (Figure 8).

This system has been used to process more than fifty different two-frame matching problems drawn from six different multi-frame sequences. All the sequences are composed of images of real scenes and contain more than one hundred lines each. No special attention was paid to the magnitude of the displacements between frames, and the system seems reasonably robust to the problems posed by undergrouping errors. Encouraged by the quality of the two-frame results, the system was run repeatedly on successive frames of a multi-frame sequence, creating a *directed acyclic graph*, or *dag*, representing the splitting and merging of line segments over time. The results of one such multi-frame experiment, involving several rotating objects, are shown in Figures 9-12. Results from a second sequence, taken by camera mounted on a mobile robot panning by a stairway, are shown in Figures 13-16.

Unfortunately, the interpretation of such a representation is non-trivial. For example, one can not tell from local information alone whether a particular split or merge in the dag is due to an undergrouping or overgrouping failure. Although the minimal bipartite cover functions well when the set of line segments in each frame is the same (except for fragmentation due to undergrouping) it performs badly when wholly new line segments appear or dissapear. This can be caused by the initial filtering operation used to reduce the number of line segments in each frame.

5. P.O. IN PARAMETER SPACE

As mentioned before, overgrouping occurs when, through chance, two tokens are juxtaposed in such a way as to satisfy the requirements for grouping. For example, two coplanar line segments, will appear colinear when viewed from any point in the plane in which they both lie (i.e. the *degenerate view plane*). Such a pair of segments is likely to satisfy the grouping requirements in an algorithm such as Boldt's, and will be grouped as a single line segment; See Figures 17 and 18. The probability of a moving sensor passing through the degenerate view plane for some pair of lines is relatively high, especially in a man made environment, due to the plethora of horizontal and vertical lines. However, if we choose to describe each line as a point in the $\rho - \theta$ parameter space, and examine the set of such points through time, *we will find two distinct trajectories that intersect during* the degenerate view. The appropriate solution seems to be to divide the set of points in parameter space into distinct trajectories corresponding to separate physical entities. This is a perceptual organization problem, and the space to be organized is defined by the parameters of the token. For point tokens, the parameter space happens to be the image plane, but for line segments, the most suitable parameters are ρ and θ, which are stable even when undergrouping errors make the determination of the segment's endpoints impossible. Solving the problems peculiar to perceptual organization of line token trajectories in $\rho - \theta - t$ will be our major goal for the next few months. By extending the "perceptual radius" to a larger number of frames, we hope to take advantage of good continuity, which will permit matching in spite of single frame grouping errors.

6. CONCLUSION

An optical flow field is a vector function of the image plane. It is a very simple characterization of the changing intensity function that results when a dynamic scene is imaged. Compared to token matching, it is a relatively well developed paradigm and several different algorithms exist for computing it. This

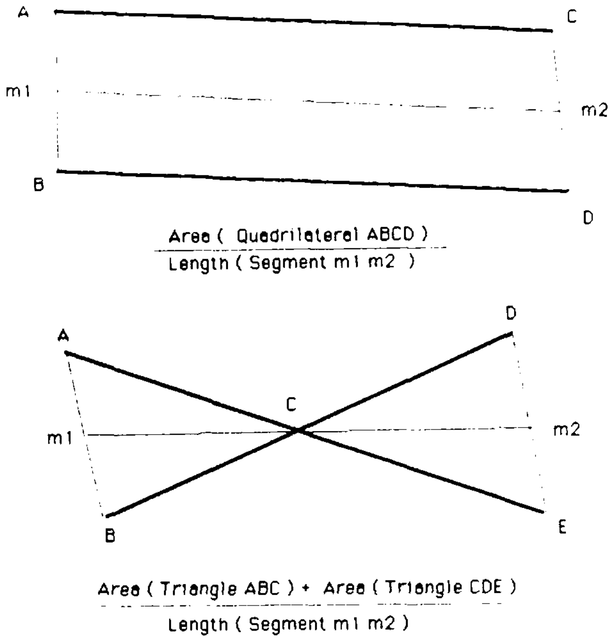

A
C
m1
m2
B
D
Area (Quadrilateral ABCD)
Length (Segment m1 m2)
D
A
C
m1
m2
B
E
Area (Triangle ABC) + Area (Triangle CDE)
Length (Segment m1 m2)

Figure 7. The positional discrepancy metric approximates the average distance between the candidate match and the predicted position of the frame two line.

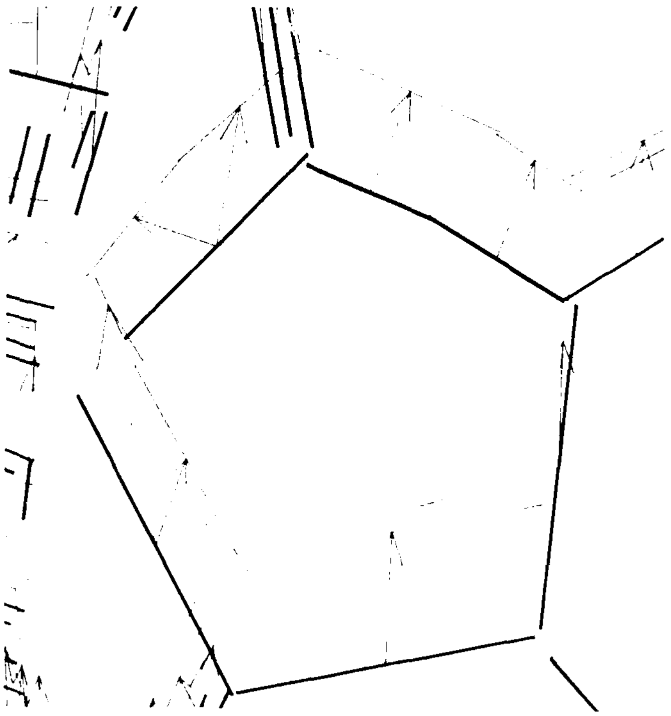


Figure 8. The *minimal bipartite cover* for each connected-component of the time-link graph. In this example, a single arc has been removed.

paper explores the possibility of translating optical flow into token matches, creating a more abstract representation based on a directed acyclic graph. The nodes of this graph are tokens corresponding to spatial structure and the arcs reflect correspondence between frames. In addition to the spatial displacement of the token, this representation makes the changing values of the token's parameters through time explicit. The approach is demonstrated by a working implementation which uses line tokens. Finally, it is proposed tnat the best path to pursue in future work is perceptual organization in the parameter space of the token. Hopefully, this will provide increased reliability in the face of single frame grouping errors.

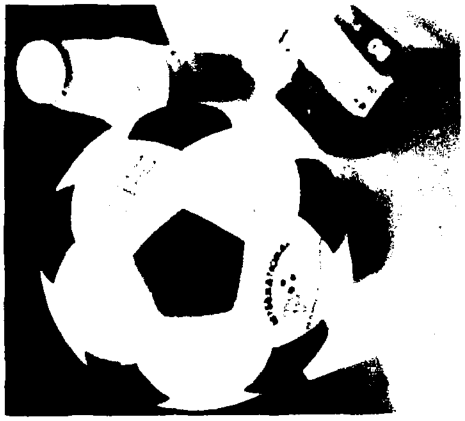


Figure 9. The first frame of a motion sequence containing multiple independently moving objects.

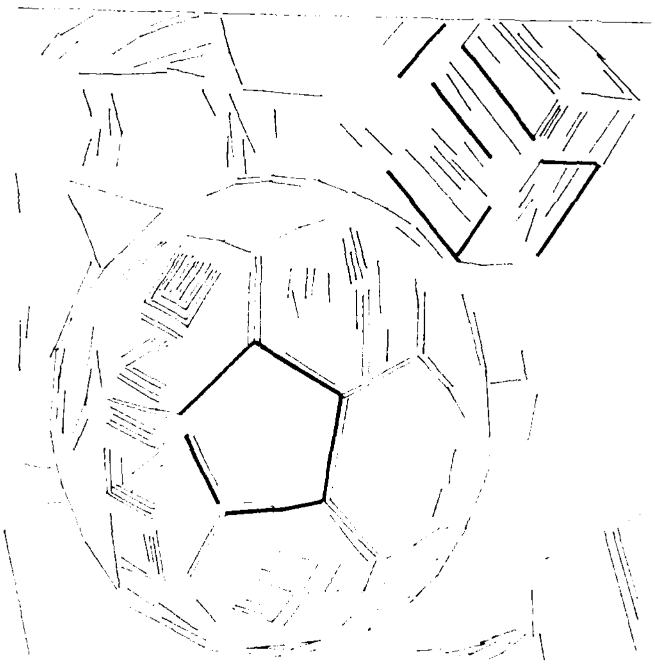


Figure 10. The line tokens computed for the first frame. Line tokens which will be used to illustrate the output of the matching process are displayed thick.

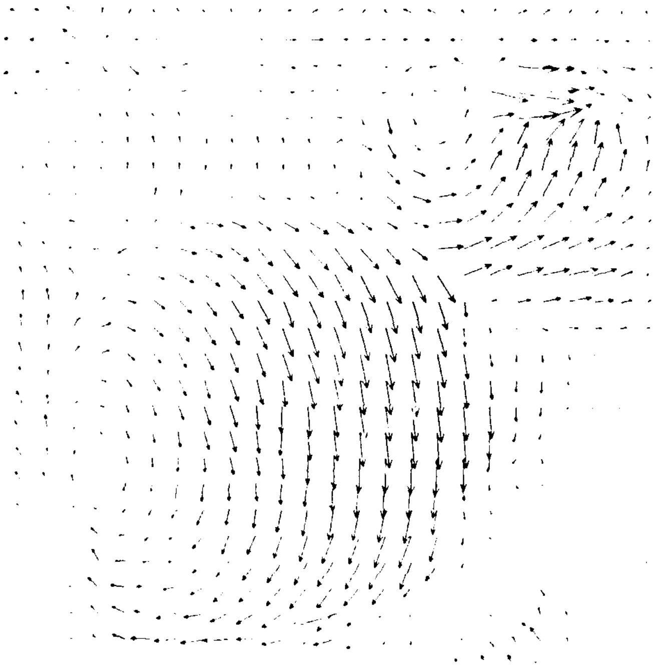


Figure 11. The displacement field computed for the first and second frame of the sequence. Note the rotation of the box and the soccer ball.

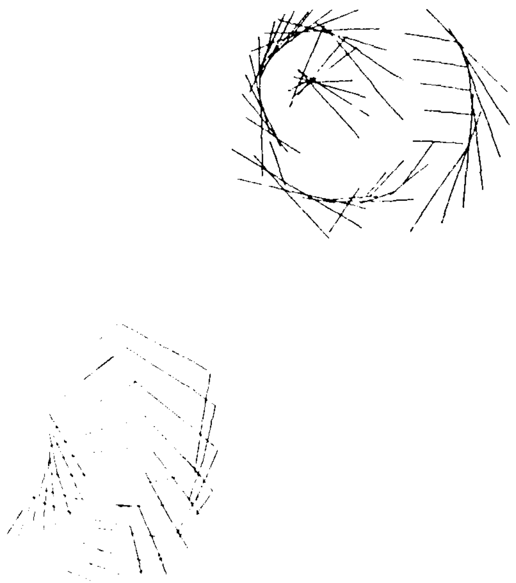


Figure 12. The output of the matching process for selected lines. This figure was computed by displaying the lines encountered during a depth first traversal of the *directed acyclic graph* representing the token matches. Note the splitting and merging of the lines outlining the panel of the soccer ball. The changing orientation of the line tokens is explicitly represented.

ACKNOWLEDGMENTS

We wish to thank Michael Boldt and P. Anandan for many valuable discussions and for providing the tools that made this work possible. Thanks also to Harpreet Sawhney and Rich Weiss.

References

[1] Anandan, P., Measuring Visual Motion from Image Sequences, Ph.D. Dissertation, COINS Dept., University of Massachusetts, Amherst, Mass, March 1987.

[2] Barnard, S.T. and Thompson, W.B., Disparity Analysis of Images, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-2, Number 4, July 1980, pp. 333-340.

[3] Besl, P.J. and Jain, R.C., Segmentation Through Symbolic Surface Descriptions, *Proceedings of IEEE Computer Vision and Pattern Recognition Conference*, Miami Beach, Fla., 1986, pp. 77-85.

[4] Boldt, M. and Weiss, R., Token-Based Extraction of Straight Lines, COINS Technical Report 87-104, University of Massachusetts, Amherst, Mass, October 1987.

[5] Glazer, F., Hierarchical Motion Detection, Ph.D. Dissertation, COINS Dept., University of Massachusetts, Amherst, Mass, January 1987.

[6] Haralick, R., Laffey, T. and Watson, L., The Topographic Primal Sketch, *The International Journal of Robotics Research*, Spring 1983.

[7] Hildreth, E.C., The Measurement of Visual Motion, Ph.D. Dissertation, Dept. of Electrical Engineering and Computer Science, MIT, Cambridge, Mass., 1983.

[8] Horn, B.K.P., and Schunck B.G., Determining Optical Flow, *Artificial Intelligence*, Vol. 17, pp. 185-203.

[9] Lowe, D., *Perceptual Organization and Visual Recognition*, Kluwer Academic Publishers, Boston, Mass., 1985

[10] Marr, D., *Vision*, Freeman Press, San Francisco, Cal., 1982

[11] Marr, D. and Ullman, S., Directional Selectivity and its Use in Early Visual Processing, *Proceedings of the Royal Society of London*, B211, pp. 151-180, 1981.

[12] Moravec, H.P., Towards Automatic Visual Obstacle Avoidance, *Proceedings of the 5th International Joint Conference on Artificial Intelligence*, 1977.

[13] Nagel, H.H. and Enkelmann W., An Investigation of Smoothness Constraints for Estimation of Displacement Vector Fields from Image Sequences, *IEEE Transactions on PAMI*, Vol. PAMI-8, pp. 565-593, 1986.

[14] Nagel, H.H., On the Estimation of Dense Displacement Vector Fields from Image Sequences, *Proc. of ACM Motion Workshop*, Toronto, Canada, pp. 59-65, 1983.

[15] Prager, J.M. and Arbib, M.A., Computing the Optic Flow: The MATCH Algorithm and Prediction, *Computer Vision, Graphics and Image Processing*, Number 24, pp. 271-304, 1983.

[16] Stevens, K. and Brookes, A., Detecting Structure by Symbolic Constructions on Tokens, *Computer Vision, Graphics, and Image Processing 37*, pp. 238-260, 1987.

[17] Weiss, R. and Boldt, M., Geometric Grouping Applied to Straight Lines, *Proceedings of IEEE Computer Vision and Pattern Recognition Conference*, Miami Beach, Fla., 1986, pp. 489-495.

[18] Williams, L.R. and Hanson, A.R., Depth From Looming Structure, *Proceedings of the DARPA Image Understanding Workshop*, Cambridge, Mass., 1988.

[19] Witkin, A., and Tenenbaum, J., What is Perceptual Organization For?, *Proceedings of AAAI '83*, pp. 1023-1026.

[20] Ullman, S., *The Interpretation of Visual Motion*, MIT Press, Cambridge, Mass., 1979.

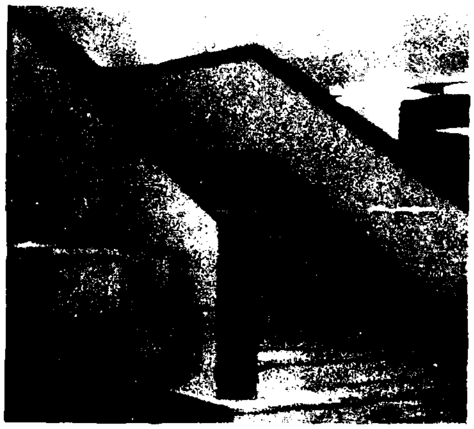


Figure 13. The first frame of a motion sequence taken by a mobile robot panning by a stairway.

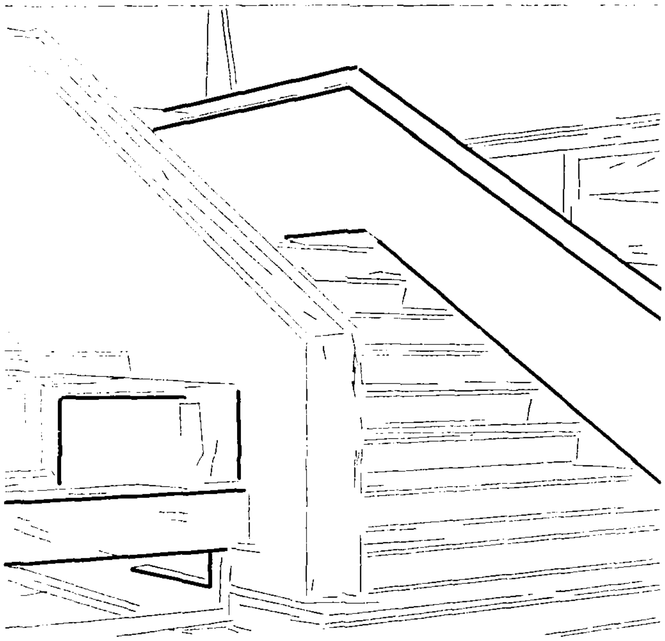


Figure 14. The line tokens computed for the first frame. Line tokens which will be used to illustrate the output of the matching process are displayed thick.

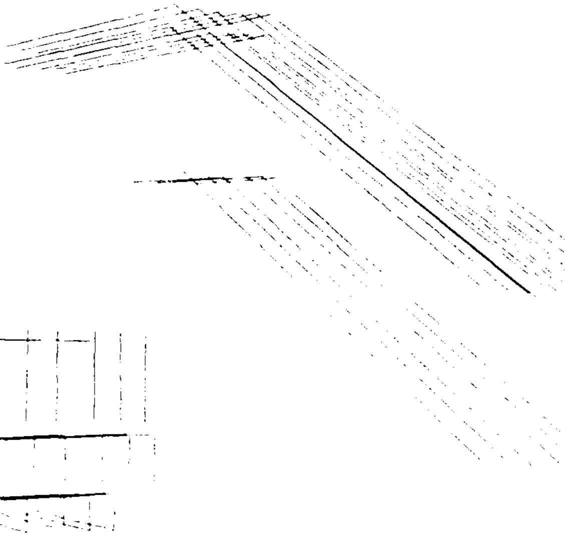


Figure 16. The output of the matching process for selected lines.

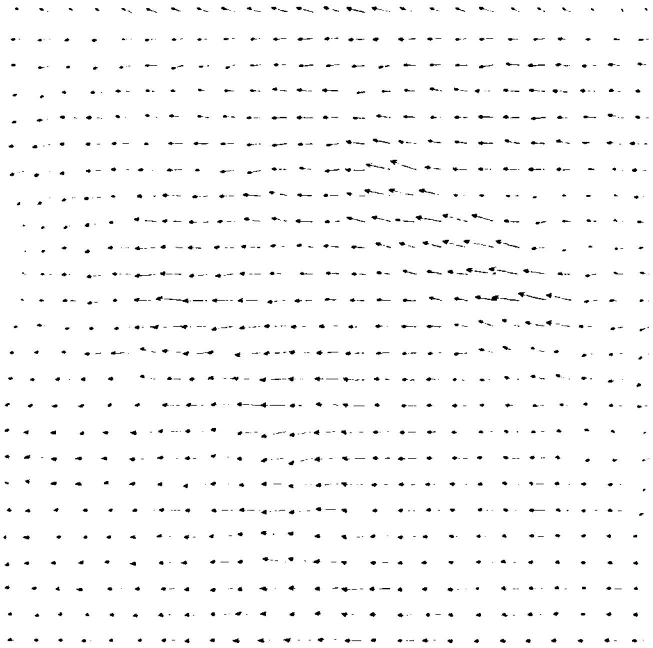


Figure 15. The displacement field computed for the first and second frame of pan sequence.

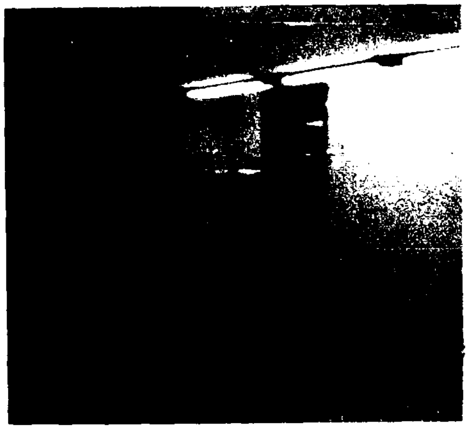


Figure 17. The fifteenth frame of a multiframe sequence depicting a *degenerate view* of line segments on the staircase and doorway. The line segments and the camera are coplanar in three space.

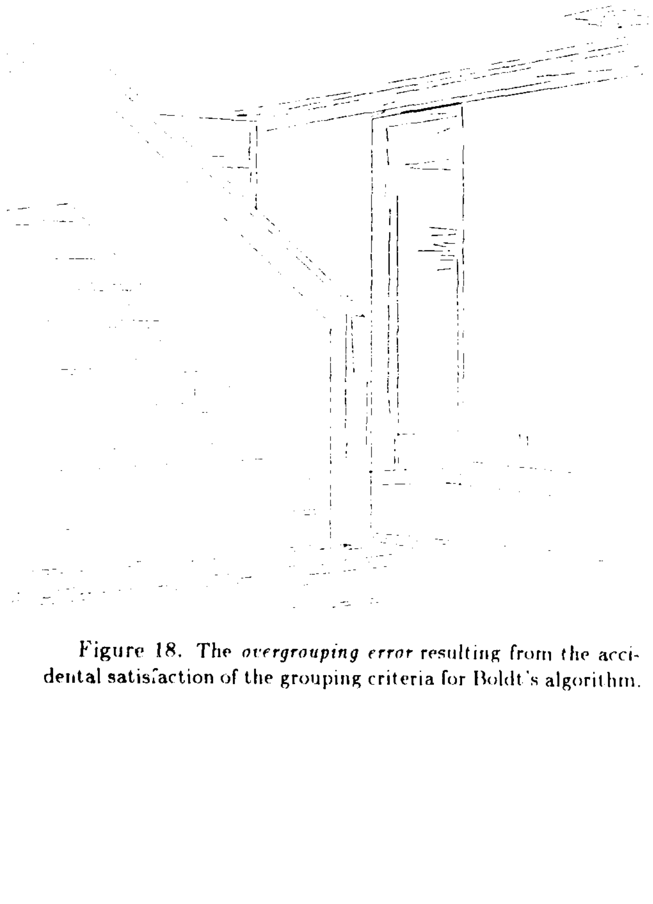


Figure 18. The *overgrouping error* resulting from the accidental satisfaction of the grouping criteria for Boldt's algorithm.

Structure Recognition by Connectionist Relaxation: Formal Analysis

Paul Cooper
Department of Computer Science
University of Rochester
Rochester, NY 14627
cooper@cs.rochester.edu

Abstract

The paper formally describes a connectionist implementation of discrete relaxation, for labelled graph matching. The application is fast parallel indexing from structure descriptions. The network is limited by complexity considerations to the detection and propagation of unary and binary consistency constraints.

The convergence of the algorithm is proved. The desired behaviour of the algorithm is formally specified, and the fact that the network correctly computes this is proved. Explicit and exact space and time resource requirements are developed.

1. Introduction

This paper provides a formal description and analysis of a connectionist network whose experimental performance was described earlier [Cooper and Hollbach 1987]. The network performs a limited kind of labelled graph matching by discrete relaxation (or constraint propagation). It was designed for the indexing task in recognizing structured objects.

The convergence of the algorithm is proved. The desired behaviour of the algorithm is formally specified, and the fact that the network correctly computes this is proved. Explicit and exact space and time resource requirements are developed. The feasibility of the algorithm given its resource requirements is discussed.

2. Graphs and Matchings

Various types of graphs and relational structures are often used to represent image information (eg. [Kitchen and Rosenfeld 1979, Shapiro and Haralick 1981]). In this section, I introduce simple definitions for graphs and matching. These particular definitions were chosen to map easily onto the corresponding connectionist network that is specified later.

Definition 1:

A *labelled graph* is a triplet G = (P,R,L), where P is a set of *parts*, or more typically, *nodes*, R is a set of *relations* or *arcs*, and L is a set of node labels. The parts (elements of the set P) are designated p_i. The label associated with p_i is designated l_i. For simplicity, the l_i are taken to be integers between 1 and 10. An arc between nodes p_e and p_f is designated r_{ef}.

Sometimes graphs and their components are designated with superscripts, to specify which graphs they belong to: eg. G^A, l_i^A, r_{ef}^A. The rest of the paper concerns itself primarily with 2 archetypical graphs:

G^A - the candidate object

G^B - the target model

Definition 2:

a *matching* is a pair of sets representing a mapping between 2 labelled graphs. The matching M^{AB} between graphs G^A and G^B is:

$$M^{AB} = (M_P^{AB}, M_R^{AB})$$

where the elements of each set are pairs that designate particular pairwise mappings. To indicate the mapping between p_i^A and p_j^B, an element of M_P is designated m_{ij}. The arc-arc mapping between r_{ef}^A and r_{gh}^B is designated $n_{ef,gh}$. Note that arc matches have no direction.

If for some p_i, there is exactly 1 m_{ij}, p_i is said to match p_j. A node match m_{ij} is called a *valid match* if $l_i^A = l_j^B$. I often refer to generic elements of P, M_P and M_R as respectively p_i, m_{ij} , and $n_{ef,gh}$, without complete quantification.

Note that a matching as defined is completely unconstrained. Matchings are not directional. It may be that a matching is not one-to-one. Matchings may also be inconsistent, where consistency is defined as follows.

Definition 3:

a node match m_{ij} and arc match $n_{ef,gh}$ are *consistent* iff:

$(i=e \mid i=f) \,\&\, (j=g \mid j=h)$

(ie. the matched nodes are each one end of the matched arcs).

An arc match is said to be locally consistent if it has consistent node matches at each end of the arc.

It is clear that a matching which is both one-to-one and locally consistent is an isomorphism between G^A and G^B.

3. Network Configuration

Next, I specify a connectionist model which can represent any 2 graphs G^A and G^B, and all possible matchings between them. Needless to say, the model will also compute locally consistent matches, including isomorphisms. I adopt the definitions of Feldman and Ballard [1982] as the formal basis for the connectionist units.

Connectionist models are inherently finite. Therefore, the size of the graphs which can be matched must be bounded a priori. Let some constant N bound the number of nodes in the graph. Aside from this size limitation, the model will be universal for any pairwise graph matching problem.

3.1 Units

I now define sets of units corresponding to all elements of the 2 graphs G^A and G^B, and all potential matchings M^{AB}. For convenience, I will refer to the individual connectionist units with the same designators as the graph and matchings elements, leaving context to disambiguate. The following are the unit sets:

$U(P^A)$ - the units corresponding to the graph nodes of the candidate object

$U(P^B)$ - likewise for the target model graph

$U(M_P)$- the node matching units corresponding to the m_{ij}

$U(R^A)$ - the units corresponding to the arcs of the candidate object

$U(R^B)$ - likewise for the target model graph

$U(M_R)$- the arc matchings units corresponding to the $n_{ef,gh}$

I call the units $U(P^A)$, $U(P^B)$, $U(R^A)$, and $U(R^B)$ *graph units*, and the units $U(M_P)$ and $U(M_R)$ *matching units*. In particular, the m_{ij} are the node matching units and the $n_{ef,gh}$ are the arc matchings units. The graph units effectively act as input units to the model, describing the graphs G^A and G^B for a particular problem instance.

These sets of units are best conceived of in terms of 2 arrays, as in Figure 1. The left-hand array represents the nodes of both graphs, and all possible node matchings. The right-hand array represents the arcs of the graphs, and all possible arc matchings. This connectionist representation of graphs and matchings is a classic example of the connectionist unit/value

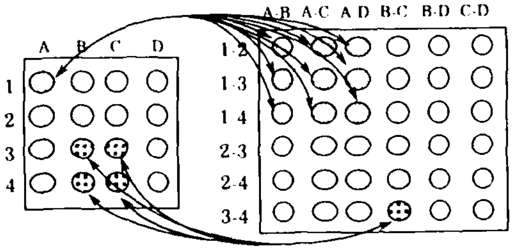


Figure 1: node matching array, arc matching array, and example constraint propagation links

principle in action. Lack of an interpreter forces the network to explicitly represent all bindings between the 2 graph component sets. If exactly one unit in each row and column of each array were active, this would represent an isomorphism between the 2 graphs.

3.1.1 Network Complexity: Units

In analyzing the complexity of a connectionist network, the most important characteristic is the number of units required for the computation. But the fixed size of the universal network is clear:

$$|U(P^{obj})| = |U(P^{model})| = N$$
$$|U(M_P)| = N^2$$
$$|U(R^{obj})| = |U(R^{model})| = \frac{1}{2}N(N-1)$$
$$|U(M_R)| = [\frac{1}{2}N(N-1)]^2$$

So the total number of units is $O(N^4)$.

3.2Connections

The connections between the units are now specified. All connections are symmetric, with weights 1.

Graph Input Connections

- all p_i^A connect to all m_{ij} for all j
- all p_j^B connect to all m_{ij} for all i

- all r_{ef}^A connect to all $n_{ef,gh}$ for all gh
- all r_{gh}^B connect to all $n_{ef,gh}$ for all ef

These connections allow the representation of the input graphs to interact with the matchings units.

Winner-take-all Connections

Each m_{ij} and $n_{ef,gh}$ is to take part in a winner-take-all competition (WTA) against the other units in its row and column. There are a variety of implementations of WTA nets, any of which is suitable. For simplicity in describing unit functions later, a WTA network is used in which each unit compares itself to the maximum of all the others.

- each m_{ij} is connected to m_{ik} for all k and m_{kj} for all k
- each $n_{ef,gh}$ is connected to $n_{ef,ab}$ for all ab and $n_{ab,gh}$ for all ab

Cross-array Connections

- each $n_{ef,gh}$ connects to exactly 4 m_{ij} as follows:

(i) $i=e \;\&\; j=g$
(ii) $i=e \;\&\; j=h$
(iii) $i=f \;\&\; j=g$
(iv) $i=f \;\&\; j=h$

These cross-array connections allow each potential arc match to connect to all 4 consistent node matches. With the p_i^A designated by numerals, the p_j^B designated by letters, and the r_{ef}^A and r_{gh}^B designated correspondingly, Figure 1 demonstrates these cross-array connections.

3.2.1 Network Complexity: Connections

Sometimes the number of connections is relevant to determining the feasibility of a network. Fan-in and fan-out values are relevant as well.

Number of Connections

- there are $2N^2$ graph input connections in the node matching array
- there are O(N4) graph input connections in the arc matching array
- there are N2 (N - 1) WTA connections in the node matching array
- there are $O(N^6)$ WTA connections in the arc matching array
- there are $O(N^4)$ cross-array connections

Aside from the WTA network (for which more efficient implementations exist), the are $O(N^4)$ connections in the network.

Fan-out

- the fan-out of the graph units is N and N^2 for the node and arc units respectively
- fan-out for the $n_{ef,gh}$ arc matching units is $2(N^2 - 1)$ for the WTA connections, and 4 for the cross-array connections

- fan-out for the m_{ij} node matching units is 2(N-1) for the WTA connections, and $(N - 1)^2$ for the cross-array connections.

Alternate WTA configurations have lower fan-out and numbers of connections. It is also possible to avoid the $O(N^2)$ fan-out from each m_{ij} across the arrays. For example, an output tree could be used.

3.3 Network Complexity: Feasibility

With an analysis of the number of units and connections required for the network, we have characterized the space complexity of the network. The question then becomes one of whether the complexity allows a feasible implementation of the network. Essentially, the network is N^4 in units and connections, and N^2 in fan-out. Usually, the feasibility of connectionist networks is determined by comparing the space complexity required with what is available in the human brain. Except for large N, by this comparison the space complexity of this network is quite small. This is particularly true if one recalls that the algorithm is not designed for matching general graphs of arbitrary size, but for matching structured representations of objects. In creating a structured representation for objects, sensibility and complexity bounds (like the ones here) suggest the use of hierarchies, which keeps N quite low. (By quite low, I mean around 10. Structured representations with many more than 10 pieces would be quite amenable to hierarchic representation.)

An alternative way to verify that the space complexity requirements of the network are reasonable is to attempt to actually implement it. As the application experiment has shown [Cooper and Hollbach 1987], this is indeed feasible. Even for experiments with larger N, say N about 10, it is quite straight-forward to implement a 10,000 unit network. (Such a network implemented on a parallel processor such as the Butterfly can even be expected to be simulated in reasonable time [Fanty and Goddard 1985]).

4. Network Computation

4.1 Overview

With the background and network configuration established, it is now possible to describe in detail how the network performs its relaxation and correctly matches the 2 graphs.

Clearly, the 2 arrays of matching units represent all possible individual match pairings, as well as all combinations. Each particular match for a given graph element (node or arc) competes with all other possibilities. The reinforcement of locally consistent node and arc matches by propagation of constraints between the 2 networks ultimately determines the final matching between the graphs.

There are 2 initialization steps. First, the particular case of G^A and G^B to be computed is instantiated by locking the potential of the input graph nodes. Second, the matching units compute from the inputs their initial state. It is at this stage that match *seeds are determined.* That is, local uniquenesses are found in both graphs and matched. Such bound "winners" have high potential.

The relaxation itself is a synchronous computation. Each time step consists of 2 substeps. The first substep is the constraint propagation phase. In this substep, node matching units send messages to consistent arc matching units, and vice versa. The second substep is the winner-take-all contest. Units which win the contest signify matches between particular nodes or particular arcs. Losers signify incorrect matches; these units eventually turn off. As the computation proceeds, the high ("matched") potential propagates bach and forth, growing larger and larger locally consistent matches from the seed matches. The match terminates when the largest possible number of locally consistent matches have been determined.

4.2 Internal Unit Structure

4.2.1 Graph Units

Graph units, including the p_i^A p_j^B r_{ef}^A and r_{gh}^B, are very simple. They each have a set of discrete states {*off, on*}. The potential corresponding to state *off* is 0 for the p_i and r_{ef}, while the on potential is 1 for the r_{ef} and equal to $l_i$for the p_i. Unit output is equal to unit potential.

These units are locked in one state (and potential) or the other for the duration of the computation. Which units are active depends on the particular instance of G^A and G^B, as described in the next section.

4.2.2 Matching Units

The units that represent particular matches are more complex. Each unit u_i, where $u_i \in M_P$ or $u_i \in M_R$ (ie. the u_i is a m_{ij} or $n_{ef,gh}$) is described as:

$u_i = <S_i, X_i, D_i>$

Each is a tuple consisting of

S_i - the unit's state, where $S_i \in$ { *off, contending, bound* }

X_i - the unit's potential (equal to its output), where $X_i \in \{0, 1, 2, \ldots 20\}$

D_i - a set of data inputs to the unit

The data inputs D_i are conveniently divided into sites, with a vector of inputs at each site. Thus:

$$D_i = \{D_{graph}, D_{WTA}, D_{cross\text{-}array}\}$$

corresponding to the 3 types of connections attached to each matching unit. Note that the same subscript is used to designate the unit, state, and potential. Thus, the state of a typical node matching unit $m_{ij}$is designated simply S_{ij}.

4.3 Problem Instantiation

The graph units p_i^A, p_j^B, r_{ef}^A, and r_{gh}^B represent all possible object and model graphs (of size bound N). These units act as the input to the match computation. They must first be activated correctly to represent the particular candidate object graph and target model graph for this computation. This process occurs as follows.

For each node in the object graph G^A, the corresponding node unit p_i^A is activated. The potential of the unit is locked at l_i, the label of p_i. Units p_j^B are activated likewise.

Suppose the input graphs have N^A and N^B nodes respectively. Note that they may be unequal, and likely less than N. Only N^A nodes p_i^A are activated.

For each arc present in the graphs, an arc unit r_{ef} is activated. Because the arcs are unlabelled, the potential of these units is locked to 1. The mapping of graph arcs to arc units must correspond in the obvious way to the mapping of nodes to node units. Note that for most problem instances (except for those with very highly connected graphs), many fewer arc units are activated than exist in the general-use network.

4.4 Initialization of Matching Units

Match units receive data D_{graph} from the graph input units initialized as described above. However, it is only in the very first time step that the potential of the match units changes as a function of these graph inputs D_{graph}.

Activation Function for Initialization of Node Match Units

For each unit m_{ij}, 2 specific data inputs of D_{graph} are identified and designated as d_{p_i} and d_{p_j} ; ie. the activation coming from p_i^A and p_j^B. In pseudocode, the initialization activation function is then:

```
if initialization step then
    if $d_{p_i} = d_{p_j}$ then
        $S_{ij} \leftarrow$ contending
        $X_{ij} \leftarrow 1$
    else
        $S_{ij} \leftarrow$ off
        $X_{ij} \leftarrow 0$
```

Activation Function for Initialization of Arc Match Units

For the arc matching units $n_{ef,gh}$, the data inputs $d_{r_{ef}}$ and $d_{r_{gh}}$ are special.

```
if initialization step then
    if $d_{r_{ef}} = d_{r_{gh}}$ then
        $S_{ef,gh} \leftarrow$ contending
        $X_{ef,gh} \leftarrow 1$
    else
```

```
$S_{ef,gh} \leftarrow$ *off*
$X_{ef,gh} \leftarrow 0$
```

Note that while the activation functions for the units are essentially the same, their effect is different. In the node matching array, all units representing *valid* matches are initialized to the *contending* state, because the input represents the labels on the p_i and p_j. In the arc matching array, all possible arc matches (corresponding to arcs which actually exist in this problem instance) are activated.

4.5 Unit Function

The state and potential of a unit are generally functions of the previous state, previous potential, and current input. This section describes the unit function - the piece of code which determines what the state and potential of a unit will be, given the previous state and potential, and the current inputs. Connectionist nets typically have large numbers of units whose state is controlled by the same function. In the graph matching network, it is convenient for proof purposes that the node matching units and arc matching units have slightly different unit functions.

Both unit functions access a time state variable as well as previous state and a subset of current inputs. Strictly speaking, this is just an approximation to what would actually be required, which would be a timing network to keep the units in synchrony, with the appropriate substeps.

The unit functions also use a distinguished potential to signify their being in the state *bound* - this potential is denoted β, and is equal to an activation of 5.

Unit Function for Node Matching Units u_i, where u_i is a m_{ij}

```
if $S_i$ = contending
    if winner-take-all substep then

        if $X_i > \max(D_{WTA})$ then
            $S_i \leftarrow$ bound
            $X_i \leftarrow \beta \leftarrow 5$
        else
            if $X_i < \max(D_{WTA})$ then
                $S_i \leftarrow$ off
                $X_i \leftarrow 0$
            else
                $S_i \leftarrow$ contending
                $X_i \leftarrow 1$


    else if propagation substep then
        $X_i \leftarrow \Sigma d$ where $d \in D_{cross\text{-}array}$
```

Unit Function for Arc Matching Units u_i, where u_i is a n_{efgh}

```
if $S_i$ = contending
    if winner-take-all substep then

        if $X_i > \max(D_{WTA})$ then
            $S_i \leftarrow$ bound
            $X_i \leftarrow 1$
        else
            if $X_i < \max(D_{WTA})$ then
                $S_i \leftarrow$ off
                $X_i \leftarrow 0$
            else
                $S_i \leftarrow$ contending
                $X_i \leftarrow 0$

    else if propagation substep then
        $X_i \leftarrow \Sigma d$ where $d \in D_{cross\text{-}array}$
```

The unit functions are basically straight-forward. One complication arises because they are not identical. The differences between them are subtle but useful at the proving stage. The arc matching nodes in their *contending* states output a potential of zero, and in their *bound* state a potential of 1. The node matching units receiving these messages can thus only discriminate between *bound* and not *bound* states. The node matching units send 0, 1, and β, on the other hand, so the receiving arc matching units can distinguish between *off*, *contending*, and *bound* messages.

5. Network Properties

5.1 Convergence

Theorem 1: The Network Converges

Proof:

Clearly, only the matching units m_{ij} and $n_{ef,gh}$ are relevant. At the completion of each time step, each such unit u_i has state $s_i \in \{$ *off, contending, bound* $\}$. Define a global goodness measure or total for the network

$T = \Sigma_i T_i$

as the sum of the individual unit goodnesses, where

$T_i = 1$ if $S_i =$ *bound*

$T_i = 0$ if $S_i =$ *off*

$T_i = -1$ if $S_i =$ *contending*

Now, units in state *bound* and state *off* cannot change state. Therefore, if a unit u_i changes state, T_i must increase. Therefore, T is a monotonically increasing function.

But the number of units is a finite constant, so T can only increase a finite number of times.

Therefore, the computation must eventually converge.

QED

5.2 Correctness

The network works by the detection and propagation of local constraints. Because of the nature of the network's representation of matchings, the only local constraint available is consistency between valid (label agreeing) node matches and arc matches. With this in mind, it is necessary to characterize formally just what kinds of matches the network can be expected to find, and then to prove that it does indeed find them. This is the subject of the next few sections.

We wish the network to determine matchings that are locally consistent. The very nature of the algorithm is based on this idea. But further, we want the network to detect the matching with minimum ambiguity, given its inherent limitations as an algorithm. Our network can only resolve matching ambiguities based on local information. Some ambiguities are only resolvable with non-local information. I refer to a matching with locally resolvable minimum ambiguity as a matching in which all graph elements that are *matchable with local consistency* are assigned unique matches.

Hypothesis:

The network determines a matching which:

a) is locally consistent
b) has locally resolvable minimum ambiguity (all nodes and arcs which are matchable with local consistency are assigned unique matches)

Some graph elements are matchable with local consistency by being locally unique. When such local uniquenesses are matched, I call this a match seed. Other nodes and arcs are matchable only after consistency has propagated to them. I consider first the seed matches. All these terms will be given tight definitions below.

5.2.1 The Network Matches Locally Unique Nodes and Arcs

Recall during the following discussion that the network's initial processing proceeds as follows:

Time Step 0:
Initialization of all m_{ij} and $n_{ef,gh}$ (cf. Section 4.1)

Time Step 1: (a) Propagation Phase:
- matching units change potential as a function of cross-array connected unit activation

Time Step 1: (b) WTA Phase
- units enter WTA competition and change potential and state depending on outcome

Time Step 2: (a) Propagation Phase
(b) WTA Phase

etc.

Definition:

a node p_e^A has a *unique unary (node) match* m_{eg} to node p_g^B iff

(i) $l_e^A = l_g^B$

& (ii) $\forall p_f^A \; [(e \neg = f) \supset (l_e^A \neg = l_f^A)]$

& (iii) $\forall p_h^B \; [(g \neg = h) \supset (\; l_g^B \neg = l_h^B)]$

In the candidate object graph, there some node with a unique label (ie. no other node in the candidate object has that label). In the target model graph, some node has the same label, and that label is unique in the target model as well.

Theorem 2: The network detects unique unary (node) matches.

Proof:

After time step 0, consider node matching unit m_{ij}. If $p_i^A = p_j^B$ is a unique unary label match, then $S_{ij} =$ *contending* and $P_{ij} = 1$. But also:

$\forall x, m_{xj} \; [(x \neg = i) \supset (S_{xj} = \mathit{off} \;\&\; P_{xj} = 0)]$

& $\forall y, m_{iy} \; [(y \neg = j) \supset (S_{iy} = \mathit{off} \;\&\; P_{iy} = 0)]$

In other words, m_{ij} is uniquely active in its row and column.

The first (propagation) substep of Time Step 1, nothing is changed. So, in the second (WTA) substep of Time Step 1:

$\forall x, m_{xj} \; [(x \neg = i) \supset (X_{ij} > X_{xj})]$

& $\forall y, m_{iy} \; [(y \neg = j) \supset (X_{ij} > X_{iy})]$

Therefore, at the end of Time Step 1, $S_{ij} =$ *bound.*

QED

Definition:

an arc $r_{ef}^A \; = (p_c^A, \; p_f^{A)}$ has a *unique binary (arc) match* $n_{ef,gh}$
to arc $r_{gh}^B \; = (p_g^B, p_h^B)$ iff

(i) $l_e^A \neg = l_f^A \;\; \& \;\; l_g^B \neg = l_h^B$
(the node labels on the arcs are not equal)

& (ii) either $[(l_e^A = l_g^B) \;\&\; (l_f^A = l_h^B)]$
or $[(l_e^A = l_h^B) \;\&\; (l_f^A = l_g^B)]$
(the label pairs on the ends of the arcs match in one direction or the other)

& (iii) $\forall r_{wx}^A = (p_w^A, p_x^A)$
$[(ef \neg = wx) \supset$
$(\; \neg(l_e = l_w \;\&\; l_f = l_x) \mid$
$\neg(l_e = l_x \;\&\; l_f = l_w))]$
(the label pair in the candidate object graph is unique)

& (iv) $\forall r_{yz}^B = (p_y^B, p_z^B)$
$[(gh \neg = yz) \supset$
$(\; \neg(l_g = l_y \;\&\; l_h = l_z) \mid$
$\neg(l_g = l_z \;\&\; l_h = l_y)) \;]$
(the label pair in the target model graph is unique

Theorem 3: The network detects unique binary (arc) matches.

Proof:

Consider each $n_{ef,gh}$ after Time Step 1a, the propagation phase. Recall that each $n_{ef,gh}$ is connected to exactly 4 m_{ij}; m_{eg} m_{eh} m_{fg} m_{fh}. At the beginning of time step 1, each of these m_{ij} had a potential of 1 (if $l_i = l_j$) or 0. Therefore, the potential X of any $n_{ef,gh}$ is the number of contending valid node-node matches which are consistent with the arc-arc match $n_{ef,gh}$.

Now, suppose $n_{ef,gh}$ is a unique binary match. Recall from condition *(i)* of the definition of unique binary match that the nodes on the ends of the matched arcs may not be the same. Therefore, $X_{ef,gh} \neg = 4$. To have $X_{ef,gh} = 3$ is impossible. So, $X_{ef,gh}$ must be 0,1, or 2. If $n_{ef,gh}$ is a unique binary match, then $X_{ef,gh} = 2$, and:

either $X_{eg} = 1 \;\&\; X_{fh} = 1$
or $X_{eh} = 1 \;\&\; X_{fg} = 1$ (from condition (ii) of the definition).

But because the particular combination of pairs is unique by definition, all the other elements of the row and column containing $n_{ef,gh}$ have $X < 2$ (from condition (iii) and condition (iv) of the definition).

Therefore, the potential of $n_{ef,gh}$ is larger than that of all units in its row and column, so it wins the WTA competition.

And, at the end of Time Step 1, $S_{ef,gh} =$ *bound.*

QED

Overall then, the network computes seed matches for either unique unary or binary seed matches. But what if the graphs contain no local uniquenesses to act as match seeds? Such graphs are highly ambiguous locally, containing many symmetries. The network specified cannot determine unique matches in this case, but will represent all the ambiguous matches, with matching units stuck in state *contending.*

5.2.2 The Network Propagates Matches When Possible

Definition:

an arc r_{ef}^{A} is *matchable by propagation at time t* if

(i) $\exists\, r_{gh}^{B}\, [S_{ef,gh} = \mathit{contending}]$

and either

(ii) $(S_{eg} = \mathit{bound}\ \&\ S_{fh} = \mathit{bound})\ |$
$(S_{eh} = \mathit{bound}\ \&\ S_{fg} = \mathit{bound})$
(both ends are consistently matched, in one direction or the other)

or (iii) 1 of 4 analogous situations is true. One of the 4 cases is described as follows (the others are obtained from the other 4 consistent permutations of the e,f,g, and h parameters):
$(S_{eg} = \mathit{bound})\ \&$
$(S_{fh} = \mathit{contending})\ \&$
$\forall x\, [\, (x \neg= h\ \&\ \exists\, r_{gx}^{B}\,) \supset (S_{fx} = \mathit{off})\,]$ (*)
(One end is matched, the other end has a consistent contending match, and the matched end has no symmetry with respect to propagation. In other words, p_g^{B} is connected to *exactly 1 node* p_h^{B} which is a consistent contending match with r_{ef}^{A}).

In the following proofs (Theorems 4 and 5), suppose the algorithm terminates after K time steps. In other words, $T^{K} = T^{K-1}$. In the following, I sometimes refer to "matchable by propagation at time t" by simply, "matchable".

Theorem 4: If an arc r_{ef}^{A} is matchable by propagation at time t and the network is still active (ie. $T^{t} \neg= T^{t-1}$) then the network assigns that arc a unique consistent match at the end of time t+1

Proof:

Case 1 (condition (ii) of definition above is true):

From the network topology, $X_{ef,gh} = X_{eg} + X_{eh} + X_{fg} + X_{fh}$ after the propagation substep beginning time step t+1. From condition (ii), $X_{ef,gh} = 2\beta$. But $X_{ef,gh}$ is the largest potential in its row:

$\forall\, x,y\, [\, \neg(x = g\ \&\ y = h) \supset (x_{ef,gh} > X_{ef,xy})\,]$

because $X_{ef,xy} = X_{ex} + X_{ey} + X_{fx} + X_{fy}$. But at most one of these terms is β (if x=g or y=h) and the rest must be zero, because p_e is bound to p_g and p_f is bound to p_h. $X_{ef,gh}$ is also the largest potential in its column, by an analogous argument.

Therefore, after the WTA substep, $S_{ef,gh}$ = *bound.*

Case 2 (condition (iii) of definition above is true):

Only the example case is proved. The other 3 analogous cases are identical, with systematically varied subscripts.

Again $X_{ef,gh} = X_{eg} + X_{eh} + X_{fg} + X_{fh}$ after the propagation substep beginning time step t+1. But $X_{eg} = \beta$ and $X_{eh} = X_{fg} = 0$ therefore. Also, $X_{fh} = 1$. Overall, $X_{ef,gh} = \beta + 1$. But this potential is greater than all others in the row:

$\forall\, x,y\, [\, \neg(x = g\ \&\ y = h) \supset (x_{ef,gh} > X_{ef,xy})\,]$

because $X_{ef,xy} = X_{ex} + X_{ey} + X_{fx} + X_{fy}$ and:

(i) suppose x=g or y=h. Then 1 term is β, say X_{eg} for x=g, without loss of generality. Then X_{ey} is certainly zero. But $y \neg= h$, so both the other terms must be zero as well, from the implication (*) in condition

(ii) suppose neither x=g nor y=h. Then $X_{ex} = X_{ey} = 0$. Because S_{fh}=*contending*, $X_{fx} <= 1$ and $X_{fy} <= 1$. So $X_{ef,xy} < \beta + 1$.

Analogously, the potential $X_{ef,gh}$ is greater than all others in the column.

Therefore, by the completion of the WTA at the end of time step t+1, $S_{ef,gh}$ = *bound.*

QED

Definition:

a node p_e^{A} with label l_e^{A} is *matchable by propagation at time t* if

(i) $\exists\, p_g^{B}[\, S_{eg} = \mathit{contending}\,]$
(p_e^{A} can match some node)

& (ii) $\exists\, r_{ef}^{A}, r_{gh}^{B}[\, S_{ef,gh} = \mathit{bound}\ \&$
$(\, (S_{fh} = \mathit{bound})\, |\, (l_e^{A} \neg= l_f^{A})\,)\,]$
(at least 1 connected and consistent arc

match exists, with either the other end matched, or dissimilar labels on the ends of each arc)

& (iii) $\forall\ r_{ex}^{A}\ \exists r_{yz}^{B}\,[\,(x \neg= f\ \&\ S_{ex,yz} = \textit{bound}) \supset$
$(y = g\ \&\ z \neg= h) \mid (z = g\ \&\ y \neg= h)\,]$
(for all the other matched arcs connected to e, their matches are consistent with the match of ef to gh)

& (iv) $\forall\ r_{gx}^{B}\ \exists r_{yz}^{A}\,[\,(x \neg= h\ \&\ S_{gx,yz} = \textit{bound}) \supset$
$(y = e\ \&\ z \neg= f) \mid (z = e\ \&\ y \neg= f)\,]$
(for all the other matched arcs connected to g, their matches are consistent with the match of ef to gh)

Theorem 5: *If a node* p_e^A *is matchable by propagation* at time t and the network is still active (ie. $T^t \neg= T^{t-1}$), then the network assigns that node a unique consistent match at the end of time t + 1.

Proof:

If S_{fh} is *bound*, it must be that S_{eh} is *off*, as is S_{fg}. Alternately, S_{eg} is *contending* and $l_e^A \neg= l_f^A$, so again S_{eh} and S_{fg} are *off*.

Now, after propagation $X_{eg} = \Sigma_{\alpha\sigma} X_{e\alpha,g\sigma}$. Furthermore, X_{eg} is the largest potential in its row:
$\forall y\,[\,(y \neg= g\ \&\ y \neg= h) \supset X_{eg} > X_{ey}\,]$
because:

(a) $S_{ef,gh}$ = *bound*, so X_{eg} receives an input of 1 from $n_{ef,gh}$ and no other element in the same row as m_{eg} receives an input for $S_{ef,gh}$ (S_{eh} would, but as we just saw, from condition (ii) of the definition, S_{eh} = *off*).

(b) If some other m_{ey} receives an input, then some $n_{ex,yz}$ is in state bound as well. (In other words, node e has another bound arc connected to it). But, by the consistency condition (iii), if $S_{ex,yz}$ = *bound*, then either y = g or z = g. In either case, X_{eg} always receives an input of 1 as well.

Therefore, taken together, (a) and (b) dictate that X_{eg} is always at least 1 greater than all other X_{ey} in the same row.

An analogous argument using condition (iv) of the definition holds true for all units in the same column as X_{eg}.

Therefore, X_{eg} is larger than all other units in its row and column following the propagation substep.

Therefore, after the WTA substep, S_{eg} = *bound*.

QED

5.2.3 The Network Detects a Good Match

Finally, we must prove that the network succeeds in matching *all* the matchable graph elements, both nodes and arcs, as far as consistency and symmetry will allow.

Definition:

the set of graph elements *matchable by propagation* is:
$\cup_{t=0,x}$ *nodes and arcs matchable by propagation*(t)

Definition:

the set of graph elements matchable with local consistency includes

(i) nodes with a unique unary match
& (ii) arcs with a unique binary match
& (iii) graph elements matchable by propagation

Theorem 6: The network assigns a unique consistent match to every graph element that is matchable with local consistency.

Proof: (Contradiction)

Assume the network converges in K steps, so $T^K = T^{K-1}$. Assume that there exists a matchable graph element which is not assigned a match. By Theorems 2 and 3, the network assigns matches for the unary and binary local uniquenesses, so it cannot be these. By Theorems 4 and 5, the network matches those elements which are matchable by propagation at time t < K (where the algorithm terminates after K steps with $T^K = T^{K-1}$). Therefore, if a matchable graph element exists that is not matched, it must be matchable by propagation for some t > K-1, say t = C. To reach this state at t = C (which is different from the state at t = K-1), the network must change state during every timestep from t=K-1 to t=C. But $T^K = T^{K-1}$, and the network does not change state after t=K-1.

QED

5.3 Ramifications

5.3.1 Time Complexity

There are some interesting ramifications of the correctness characteristics of the algorithm. First, it is possible to show that the network converges in a number of time steps linear in the number of graph elements being matched.

Suppose we designate N_n^A and N_a^A as the number of nodes and arcs respectively in graph A, and designate N_n^B and N_a^B the same way. Now call $SIZE^A = N_n^A + N_a^A$, and similarly for graph B. In the following proof, I also use the concept of a *propagation path.* Consider some node which is matchable by propagation at time t. Then there exists one particular associated seed match from which the match propagated to the node. Furthermore, there exists some path from that seed match to the node along which the match propagated. Finally, if both nodes and arcs are counted in the path length (but each is counted at most once, if the path contains a cycle), the length of the propagation path from the associated seed match to the node must equal exactly t.

Theorem 7: The network converges in K time steps, where K <= min($SIZE^A$, $SIZE^B$).

Proof:

Consider the smaller of the two graphs if they have unequal sizes, and suppose without loss of generality that it is graph A. The maximum length of a propagation path in graph A is $SIZE^A$. Therefore, the maximum time it can take for the seed match to propagate to worst case node match is $SIZE^A$ steps. Furthermore, propagation only occurs when matches are being made. Therefore, while graph B might have a longer path, propagation can only occur for the shorter length of time.

QED

One of the interesting aspects of this result is what it is not. That is, one might expect that the longest propagation path would be equal in length to the longest path between any 2 nodes in the graph (the diameter of the graph). This is not true, however, because propagation depends not only upon the path taken, but also upon *when* the propagation steps occur and whether or not symmetries are present at a given time. Sometimes a node becomes matchable by a very roundabout propagation path.

5.3.2 Isomorphisms and Near Matches

Suppose $N_n^A = N_n^B$ and $N_a^A = N_a^B$. If all the nodes and arcs of G^A are matchable with local consistency, the matching computed by the network is the unique isomorphism.

But suppose the sizes are equal, but more than one isomorphism exists between the graphs. Then, not all the nodes and arcs of the graphs are matchable with local consistency. But what does this mean? Well, for this to be true, there must exist some kind of local symmetry in the graph, beyond which the constraint of consistent matching cannot propagate. In this situation, all (ambiguous) isomorphic matches between the graphs are represented in the final state of the network, by competing matching units whose state has stabilized at *contending.* Clearly, if necessary, case analysis could be used to further disambiguate.

Finally, we must consider what happens if the graphs are completely dissimilar. That is, either their sizes differ, or their structure differs, or both. It is still true that the network computes all seed matches, and propagates each match as far as consistently possible. Thus, all elements matchable with local consistency are assigned matches. The final matching has the largest possible locally isomorphic subgraphs matched (with each containing a seed match).

6. Discussion and Conclusions

The central contribution of this paper has been to describe a connectionist implementation of discrete relaxation for labelled graph matching, and prove the correctness of its operation. The general utility of discrete relaxation is well known, and the abstract form of the algorithm is well understood. But the specifics of the connectionist implementation illuminate some interesting issues.

First, with the connectionist implementation the complexity of the algorithm can be made very precise. The exact space and time requirements of the algorithm have been specified. The explicit nature of the implementation makes extending the algorithm in parallel trivial, and its resource requirements remain explicit and easy to compute. The theoretical speed which one expects to obtain with massively parallel implementations of algorithms is provably present. With many implementations of relaxation (eg. [Kitchen and Rosenfeld 1979]) the time and space requirements of the algorithm are not specified.

Second, it is interesting to observe how working within the connectionist paradigm constrains a design. The requirement that there be no interpreter means each relevant value must be represented by a unit. This suggests in the context of the graph matching problem, that representing matchings for ternary or higher relations would become prohibitively expensive. Furthermore, bandwidth limitations on the messages the units can send each other (inspired by neural communication bandwidth limitations) restrict the character of what can be computed with unary and binary constraints. For example, it would be difficult to modify the algorithm so that more than one unary predicate could be used (like [Kitchen and Rosenfeld 1979] do). Likewise, bandwidth limitations restrict the potential utility of the unary predicate itself. Interestingly, the overall resource requirements of the algorithm suggest an upper bound on the number of parts a structure representation can have before a more efficient representation such as a hierarchy is required. (Structure representations with more than a small constant number of parts, like in the small double digits, would become unwieldy.)

The paper contains somewhat stronger and more specific results about what can be matched than are typically found (cf. the definitions of matchable by propagation for nodes and arcs). These results were obtainable because the exact nature of the algorithm was very tightly constrained (eg. only unary and binary constraints, with propagation occurring only under very specific conditions).

The particular performance characteristics of the algorithm suit the task problem - fast indexing from structure descriptions - very well. Complete recognition (with verification) requires that a complete correspondence between object and model be established, and this algorithm is unable to do so. But it can be used to quickly reject large numbers of model candidates, so later more expensive processes can be more confidently and efficiently applied to the remaining candidates.

Finally, the implementation and proof provide a basis from which to investigate more general indexing problems. In future work, I plan to investigate indexing from structure with uncertain and inexact structure descriptions. Connectionist implementations seem to be natural hosts for such problems.

Acknowledgements

Jerry Feldman was an indispensable help throughout. Paul Chou provided invaluable and timely suggestions and criticism. Nigel Goddard and Susan Hollbach assisted in the original work.

This work was supported by the Air Force Systems Command, Rome Air Development Center, Griffiss Air Foce Base, New York 13441-5700 and the Air Force Office of Scientific Research, Bolling AFB, DC 20332 under Contract No. F30602-85-C-0008. The latter contract supports the Northeast Artificial Intelligence Consortium (NAIC). I thank the Xerox Corporation University Grants Program for providing equipment used in the preparation of this paper.

References

Ballard, D. and C. Brown. *Computer Vision*, Englewood Cliffs, N.J.: Prentice Hall, 1982.

Biederman, I. "Human image understanding: recent research and a theory", *Computer Vision, Graphics and Image Processing* Vol. 32, No. 1, pp. 29-73, 1985.

Brooks R.A. "Symbolic Reasoning among 3D models and 2D Images", *Artificial Intelligence*, vol. 17, pp. 285-345, August 1981.

Cooper, Paul R and Susan C. Hollbach. "Parallel Recognition of Objects Comprised of Pure Structure", Proceeding DARPA Image Understanding Workshop, L.A., February 1987.

Fanty, M. and N. Goddard."Users manual: Rochester *connectionist simulator*", Computer Science Department, University of Rochester, 1986.

Feldman, Jerome A. "Four Frames Suffice: A provisionary model of vision and space." *The Brain and Behavioural Sciences 8*, 265-289, 1985a .

Feldman, Jerome A. "Connectionist Models and Parallelism in High Level Vision", *Computer Vision, Graphics, and Image Processing 31*, 178-200, 1985b.

Feldman, Jerome A. "Energy and the Behaviour of Connectionist Models", TR 155, Deparment of Computer Science, University of Rochester, 1985c.

Feldman, J.A. and D. Ballard. "Connectionist models and their properties," *Cognitive Science*, 6, 205-254, 1982.

Hoffman, D. and W. Richards, "Parts of recognition" in *From Pixels to Predicates*, Pentland, A., (Ed.) Norwood, N.J.: Ablex Publishing Co. 1985.

Hopfield, J.J. and D.W. Tank. "Neural Computation of Decision in Optimization Problems", *Biol. Cyber.* 52, pp 141-152, 1985.

Hummel, Robert A. and Steven W. Zucker. "On the Foundations of Relaxation Labelling Processes", *IEEE Trans. PAMI-5*, No. 3, 1983.

Kitchen, Les and A. Rosenfeld. "Discrete Relaxation for Matching Relational Structures", *IEEE Trans. Systerrs, Man, and Cybernetics, SMC-9, 12*, 869-874, 1979.

Lowe, David. *Perceptual Organization and Visual Recognition*, Boston MA: Kluwer Academic Publishers 1985.

Mackworth, Alan K., Mulder, Jan A. andWilliam S. Havens. "Hierarchical Constraint Propagation: Exploiting Structured Domains in Constraint Satisfaction Problems", *Computational Intelligence*, 1, 118-126, 1985.

Pentland, A.P. "Parts: Structured Descriptions of Shape", *Proceedings*, *AAAI-86*, 695-701, Phil., PA, Aug. 11-15, 1986

Shapiro, Linda G. and Robert M. Haralick. "Structural Descriptions and Inexact Matching", *IEEE Trans PAMI-3*, No. 5, pp. 504, 519, September 1981.

Extracting Generic Shapes Using Model-Driven Optimization [*]

Pascal Fua and Andrew J. Hanson

SRI International (Artificial Intelligence Center)
333 Ravenswood Avenue, Menlo Park, Califonia 94025

Abstract

We develop a model-driven approach for extracting generic shapes from monoscopic and stereographic imagery. Generic geometric constraints are used to obtain model candidates and to suggest missing model components in the image data. Competing scene parses are ranked using objective functions that include the cost of encoding the scene information in terms of a set of generic primitives and a measure of the geometric quality of the parse. As examples, we formulate generic models for buildings, roads, and vegetation clumps. The model definitions include edge characteristics, two- and three-dimensional geometric properties, region characteristics, and procedures for predicting and verifying missing shape components. Results are shown for a variety of aerial imagery.

1 Introduction

A basic goal of computer vision research is to develop sound theoretical approaches to the interpretation of information derived from imaging sensors. Scene descriptions should be phrased in terms of semantic models relevant to specific objectives; the process of generating such an interpretation is typically referred to as *model-based vision* [see, e.g., Brooks, 1981; Binford, 1982]. In do[illegible] highly constrained, *model-driven* approaches [illegible] Bolles and Horaud, 1986; Shneier et al., 1986] have [illegible] *to supplement passive model-based methods by predicting* and searching for missing model parts in the image.

We suggest that for model-based vision to be effective in less-constrained applications, such as the automatic delineation of cartographic features in aerial imagery, *generic models* must play a new and more dynamic role in all stages of the analysis. The components of our "generic-model-driven" approach to scene interpretation are:

- **Generic Models.** We use *generic* models for tasks such as automated cartography whose object classes cannot be fully specified in advance [see, e.g., Fua and Hanson, 1985, 1987a,b,c]. Models that are too specific lack the breadth and flexibility necessary to guide the interpretation process for most real problems.

- **Active Use of Models for Discovery and Prediction.** Each model definition includes mechanisms for generating and verifying missing-part hypotheses using image-based computation. Therefore, the models themselves play an essential role in finding appropriate candidate instances to be evaluated with the objective functions.

- **Image-Based Objective Functions.** For each model, we define an *objective function* that includes measures of the quality of the evidence for model instances in the observed image data. This function is used to select the best set of compatible model candidates to describe a scene.

In order to generate reasonable hypotheses for shape instances in the presence of noise and missing structures, *all* procedures in a feature-discovery system should be carried out with explicit reference to the characteristics of a set of models. Model-based vision cannot be effectively carried out using strictly passive techniques, where extracted shape information is passed to an interpreter that makes no further reference to the image data. "Low-level" vision tasks cannot be pursued independently without a clear concept of what modeling information will be required of them; similarly, "high-level" tasks must refer directly to the image to verify each semantic hypothesis.

Our approach employs a comprehensive theoretical approach to model-driven image understanding based on encoding a scene in terms of a model language and evaluating the result with objective functions. We also introduce practical implementation techniques that enable the successful application of the model-driven paradigm throughout the scene-parsing procedure.

2 Theoretical Foundations

The approach to model-based vision that we advocate seeks to *find scene interpretations*, phrased in *terms* of a particular model vocabulary, that optimize an objective function.

This theoretical approach would, in principle, require an exhaustive evaluation of an enormous static set of alternate interpretations. In practice, one would consider generating only a limited set of model hypotheses using the output of syntactic operators. But simply evaluating the objective function on the output of a simple low-level operator will seldom give reasonable results on complex images. The implementation must have effective procedures for producing relevant model candidates for the objective function to measure.

A candidate-generating procedure may produce many conflicting proposals; the objective function selects an optimal set of compatible hypotheses from among the competing alternatives.

To clarify these separate tasks, we have separated our discussion into two major sections: the current section on theoretical foundations, which emphasizes the abstract requirements of the *optimization approach* regardless of how the model candidates are acquired, and a subsequent section on implementation, which illustrates appropriate techniques for generating optimal model candidates while limiting undesirable combinatorics.

[*] This research was supported in part by the Defense Advanced Research Projects Agency under Contract No. MDA903-86-C-0084.

2.1 Generic Modeling

People can accurately classify instances of various object categories even though a particular instance may have a unique shape that they have never seen before. The introduction of generic shape classes is necessary to automate this human ability.

The generic models that we have found useful for analysis of real images possess the following properties:

- **Strong edge geometry.** Elementary edge or line data extractable from an image must be related in some direct and computable way to the object. Typical edge models include such characteristics as long, straight segments, uniform local curvature, and statistical signatures indicating jaggedness. If we now define an *optimal edge pixel* to be a pixel that is a maximum of the local image gradient in the direction of this gradient, the *edge score* can be taken as the ratio of the number of optimal edge pixels to the number of pixels that are not optimal. This approach amounts to counting the number of pixels that are step-edge pixels according to a standard definition [Haralick, 1984; Canny, 1986]. The resulting score provides a good measure of how well a set of edges fits the photometric data after the geometric model has been imposed [Fua and Leclerc, 1988].

- **Strong area signature.** Areas contained within a generic object should be characterizable by a computable signature, such as uniform or uniformly changing intensity values or textures. For real objects, such as parking lots with cars or roofs with chimneys, anomalies are to be expected; such anomalies are easily located and discounted, provided they constitute no more than a small fraction of the area. Our requirements can be met by the following fitting procedure: We find the peak of the gray-level histogram of the pixels within a patch, fit a plane to those pixels that lie within the peak, and compute a histogram of the deviations of those pixels from the fit. In general, the deviation histogram will exhibit a main peak and several smaller peaks corresponding to the anomalies. The *area score* is then taken to be the average number of bits needed to encode the histogram as a sum of Gaussian distributions.

- **Optional stereographic signature.** When stereographic information is available, structures in one image can be backprojected to matching structures in the other image. The quality of this match may be computed using a combination of the technique for area matching just described and the measure used by Barnard [1988]. That is, we compute the histogram of the intensity differences between pixels in the patch in one image and the pixels in the other image obtained by projection of the hypothesized three-dimensional position of the surface. The *stereo match score* is then the number of bits needed to encode the histogram of intensity differences as a sum of Gaussian distributions.

- **Predictability of incomplete structures.** Edge geometry and area signature are relatively straightforward to evaluate if we are given a perfect model candidate, but typical procedures for generating model hypotheses produce incomplete structures. We therefore provide *model-driven procedures that search for incomplete components of the model in the imagery.*

Generic modeling is thus characterized by exploitation of geometric edge constraints and photometric area properties, combined with the ability to fill in missing model components in order to generate reasonable model candidates for evaluation by the objective function.

The models that we have implemented to date—buildings, roads, and trees—are relatively simple. They are defined as circular graphs of edges that are geometrically related and can be completed to form closed contours conforming to the model geometry. The model definition therefore consists of the following components: (1) edge definition and geometric signature, (2) specification of an area's photometric signature and filtering thresholds, (3) definition of geometric relationships among edges, and (4) procedures that produce closed contours from a circular graph of related but disjoint edges. For roads and buildings, all four elements are present; for trees, which are much less constrained, the strong geometric restrictions are absent. There is no intrinsic obstacle to defining much more complex hierarchies of objects, but we have not yet implemented such definitions.

2.2 Model-Based Evidence Measures

The objective function is a combined measure of the deviation of a model candidate from the image-based evidence for the model and the geometric quality of the candidate.

We choose the image-based measure F to be the *effectiveness* of a model instance, which we define as *the difference between the number of bits needed to encode the photometry of a scene patch without the model versus the number required with the model.* In cases where the cost of encoding the model itself can be neglected, effectiveness is the number of bits of information that are saved by describing the image information in terms of the model rather than without it; effectiveness thus measures goodness of fit.

The geometric quality measure G is represented by a cost that we will argue is related to the probability of the occurrence of a model instance's geometric characteristics.

For a set of hypothesized model instances denoted by $M = \{m\}$, the overall score of a scene parse is

$$S = F - G, \tag{1}$$

where $F = \sum_{m \in M} F_m$ is the total effectiveness of the scene representation and $G = \sum_{m \in M} G_m$ is the total geometric cost of the parse. These definitions implicitly assume statistical independence of the individual model instances. For complex scenes, we should in principle express F and G in a way that accounts for the fact that complex models are far from statistically independent; buildings are correlated with roads, bridges with rivers, and so forth. For our present purposes, the independence assumption is a good approximation and makes the optimization problem computationally tractable.

Photometric Evidence Sources. The photometric effectiveness term F for each model candidate contains terms corresponding to the set $E = \{e\}$ of *evidence sources* being considered:

$$F_m = \sum_{e \in E} F_{m,e}. \tag{2}$$

We assume that we have chosen sources of evidence whose interdependence can be neglected. We will show below how the measures $F_{m,e}$ can be computed in practice in a noiseless information-theoretic coding framework.

We reiterate below the photometric evidence computations applied to each model candidate in our implementation:

- **Area Intensity Characteristics.** Find the peak of the gray-level histogram of the pixels within a patch, fit a plane to those pixels that lie within the peak, and compute a histogram of the deviations of those pixels from the fit. The evidential measure is the number of bits needed to encode the histogram of deviations as a sum of Gaussian distributions [Leclerc, 1988], i.e., $\sum_i n_i \log_2 \sigma_i + \sum_i n_i \log_2(n_i/N) + cN$, where σ_i and n_i are the width and number of pixels in each Gaussian, N is the total number of pixels, and $c = 0.5(\log_2(e\pi) + 1)$.

- **Edge Quality and Geometry.** Select edge segments matching the model geometry and compute the edge measure from the ratio of the number of optimal edge pixels to the number of pixels that are not optimal.

- **Stereographic Matching Characteristics.** Compute the number of bits needed to encode the histogram of intensity differences of stereo-matched pixels as a sum of Gaussian distributions.

Additional possible sources of evidence include texture measures, range characteristics, and continuity of motion in a motion sequence.

Geometric Evidence Sources. The geometric measure G enforces geometric constraints on the model candidates. When we are dealing with a limited number of shape candidates whose very discovery was prompted by a prejudice toward a particular model shape, all of the shape candidates are guaranteed to obey quite strictly the geometric requirements of the corresponding model. Thus it is often sufficient to invoke only the image-based photometric evidence measures described in the previous paragraph and G may be neglected. However, in scenes with substantial complexity and ambiguous photometry, it becomes necessary to include G in Eq. (1) to differentiate parses on the basis of geometric quality. Model-based constraint systems such as ACRONYM [Brooks, 1981] include elaborate capabilities for dealing with such geometrical questions; similar facilities could be incorporated into our framework by suitably tailoring G.

Typical examples of purely geometric measures of model quality that have been found useful are rectilinearity, jaggedness, compactness of the area of a delineated model element, and the absence of narrow appendages or links. Simple versions of such measures can be effected using thresholds on such quantities as relative angles between model edge components and the compactness of a delineated area.

2.2.1 Estimating the Photometric Effectiveness.

Our explicit choices for the fundamental effectiveness measures in Eq. (2) are motivated by a heuristic derivation based on information-encoding arguments [see, e.g., Leclerc 1988]. We assume first, for simplicity, that we are working with a two-dimensional model consisting only of area and edge properties. This type of model is in fact the one we have explored most extensively. Next, we show how stereographic properties can be handled. Further properties can be added using techniques similar to those employed for the stereographic evidence measure.

We begin by considering a patch of contiguous pixels in the image. Since we are going to consider both the area and the edge properties, we need to distinguish edges from other pixels in a meaningful way; to this end, we define an "edge" as a maximum of the local image gradient measured in the direction of the image gradient at the edge pixel [Canny, 1986]. Thus we need to define the following properties of an image patch:

$$\begin{aligned}
A &= \text{Area of the patch in pixels} \\
L &= \text{Length of patch boundary in pixels} \\
n &= \left\{\begin{array}{l}\text{Number of boundary pixels that are max-} \\ \text{ima of the local image gradient}\end{array}\right\} \\
\overline{n} &= L - n \\
&= \left\{\begin{array}{l}\text{Number of boundary pixels that are } \textit{not} \\ \text{local maxima of the local gradient}\end{array}\right\}.
\end{aligned}$$

If we treat the patch as though it were a random background, with no available model-based interpretation, we are led to define

$$\begin{aligned}
b &= \left\{\begin{array}{l}\text{Probability that a random pixel in the image is a} \\ \text{maximum of the local image gradient}\end{array}\right\} \\
k &= \left\{\begin{array}{l}\text{Average number of bits needed to encode the inten-} \\ \text{sity of a random pixel in the image}\end{array}\right\}.
\end{aligned}$$

The minimum number of bits required to encode a body of information is the negative logarithm of its probability [Rissanen, 1987]. Assuming now that pixel values are uncorrelated, a Huffman-encoding scheme is appropriate, and the *minimum* number of bits needed to encode the edge properties of all boundary pixels in a random background patch is $(-n \log_2 b - \overline{n} \log_2(1-b))$. The total encoding cost in bits of the model-free patch is then

$$B = kA - n \log_2 b - \overline{n} \log_2 (1 - b). \qquad (3)$$

Although both b and k can be estimated by direct computation in a single image, one technically should use an ensemble of images; in practice we choose these quantities to be heuristically determined constants.

Now let us introduce a model consisting of a photometric area signature and an edge geometry requirement. The cost of encoding the patch using the model is

$$C = mA - n \log_2 p - \overline{n} \log_2 (1 - p). \qquad (4)$$

where

$$\begin{aligned}
m &= \left\{\begin{array}{l}\text{Average number of bits needed to encode the devia-} \\ \text{tion from the model of the pixel values in the patch}\end{array}\right\} \\
p &= \left\{\begin{array}{l}\text{Probability that a boundary pixel of an actual ob-} \\ \text{ject is a maximum of the local image gradient}\end{array}\right\}.
\end{aligned}$$

Note that p requires prior knowledge of the nature of each image with respect to the goal of interpreting that image in terms of a particular model set. Therefore, it is seldom precisely computable and, like b and k, will be chosen in practice as a heuristic parameter.

The effectiveness of a model description now becomes

$$F = B - C = (k - m)A + n \log_2 \frac{p}{b} + \overline{n} \log_2 \frac{(1 - p)}{(1 - b)}. \qquad (5)$$

where the coefficients m, A, n and $\overline{n}$ are evaluated for each patch.

The measure F has a number of qualitative features that correspond well with what we need for scene evaluation. For example, we see that the value increases when the fit to the area model improves, since k is fixed and m approaches zero for constant intensities, and that the value also increases when the number of local maxima of the image gradient that are present in the region boundary increases relative to the number of boundary pixels that are not local maxima.

Equation (5) is thus a realization of the abstract objective function Eq. (2) that we use to evaluate two-dimensional scene parses.

In order to account for the presence of stereographic information, we add to the objective function the term

$$F_{\text{stereo}} = B_s - C_s = (k_s - m_s)A, \tag{6}$$

where m_s is the number of bits needed to encode the histogram of the intensity differences between corresponding pixels in the candidate stereo match, and k_s is the average number of bits required to describe the histogram of differences between two random patches. Although a reasonable value for k_s could be computed directly for each stereo pair, we will treat k_s as a heuristic constant just as we treat k.

2.2.2 Maximum Likelihood Interpretation

Minimal encoding and maximum likelihood are closely related [see, e.g., Cheeseman, 1983]. We now show how, under certain simplifying assumptions, the heuristic cost function that we optimize can be reduced to the logarithm of a maximum likelihood measure. To see this relationship, we consider the probability P that, given the evidence $E = \{\epsilon_i\}$, parsing the scene in terms of a particular set of model instances $M = \{m_0, m_i; i = 1 \ldots n\}$ is in fact correct. Here the indices i refer to model instances such as delineated objects, and m_0 represents the background model, i.e., the set of all pixels in the scene that do not belong to delineated objects and for which no evidence is obtained.

Iterating the decomposition formula $p(a, b|c) = p(a|b, c)p(b|c)$, we may express P as

$$\begin{aligned} P &= p(m_0, m_1, \ldots, m_n|\epsilon_1, \ldots, \epsilon_n) \\ &= p(m_0|\epsilon_1, \ldots, \epsilon_n)\prod_i p(m_i|m_0, \ldots, m_{i-1}, \epsilon_1, \ldots, \epsilon_n). \end{aligned} \tag{7}$$

where $p(m_i|m_0, \ldots, m_{i-1}, \epsilon_1, \ldots, \epsilon_n)$ is the probability that a hypothesized model instance m_i is correct given both the available evidence and the presence of a subset of other models.

We now make two assumptions about the nature of relationships between the models and the evidence that lead to a useful approximation for the scene-parse probability:

- **Independence of each model from th^ other evidence.** If the body of evidence ϵ_i refers to the i-th model, we assume that m_i is affected by its own evidence ϵ_i and *the presence of the other model instances* $m_{j\neq i}$, but is independent of $\epsilon_{j\neq i}$. Then

$$\begin{aligned} p(m_0|\epsilon_1, \ldots, \epsilon_n) &= p(m_0) \\ p(m_i|m_0, \ldots, m_{i-1}, \\ \epsilon_1, \ldots, \epsilon_i, \ldots, \epsilon_n) &= p(m_i|m_0, \ldots, m_{i-1}, \epsilon_i) \\ &= \frac{p(m_0, \ldots, m_{i-1}, \epsilon_i|m_i)}{p(m_0, \ldots, m_{i-1}, \epsilon_i)}p(m_i). \end{aligned}$$

- **Independence of the evidence from the other models.** We also assume that the evidence relative to model instance m_i is independent of the presence of other model instances, so that

$$\begin{aligned} p(m_0, \ldots, m_{i-1}, \epsilon_i) &= p(m_0, \ldots, m_{i-1})p(\epsilon_i) \\ p(m_0, \ldots, m_{i-1}, \epsilon_i|m_i) &= p(m_0, \ldots, m_{i-1}|m_i)p(\epsilon_i|m_i). \end{aligned}$$

Using these two assumptions, we have

$$\begin{aligned} p(m_i|m_0, \ldots, m_{i-1}, \\ \epsilon_1, \ldots, \epsilon_i, \ldots, \epsilon_n) &= \frac{p(m_0, \ldots, m_{i-1}|m_i)p(\epsilon_i|m_i)}{p(m_0, \ldots, m_{i-1})p(\epsilon_i)}p(m_i) \\ &= p(m_i|m_0, \ldots, m_{i-1})\frac{p(\epsilon_i|m_i)}{p(\epsilon_i)}, \end{aligned}$$

where we used Bayes' rule to combine terms in the last line. Thus we finally have

$$\begin{aligned} P &= p(m_0)\prod_i p(m_i|m_0, \ldots, m_{i-1})\frac{p(\epsilon_i|m_i)}{p(\epsilon_i)} \\ &= p(m_0)\prod_i p(m_i|m_0, \ldots, m_{i-1})\prod_i \frac{p(\epsilon_i|m_i)}{p(\epsilon_i)} \\ &= p(m_0, \ldots, m_n)\prod_i \frac{p(\epsilon_i|m_i)}{p(\epsilon_i)}. \end{aligned} \tag{8}$$

We now write

$$\log_2 P = F - G. \tag{9}$$

where

$$\begin{aligned} F &= \log_2 \prod_i \frac{p(e_i|m_i)}{p(e_i)} \\ &= \sum_i (\log_2 p(e_i|m_i) - \log_2 p(\epsilon_i)) \\ G &= -\log_2 p(m_0, \ldots, m_n). \end{aligned}$$

and argue that F and G are identifiable with the total effectiveness and geometric cost defined in Eq. (1).

If C_i denotes the number of bits of information needed to encode the observed deviations of the evidence source ϵ_i from the model m_i, and B_i denotes the number of bits to encode the evidence in the absence of a model, we have

$$\begin{aligned} C_i &= -\log_2 p(\epsilon_i|m_i) \\ B_i &= -\log_2 p(\epsilon_i). \end{aligned}$$

Thus the effectiveness measures defined earlier can be written as

$$F_i = B_i - C_i \tag{10}$$

and

$$F = \sum_i F_i \tag{11}$$

is now identifiable as the total effectiveness.

Next, in order to identify G with the cost in Eq. (1), we assume that we may express the joint probability term in Eq. (8) as

$$\begin{aligned} p(m_0, \ldots, m_n) &= k\prod_{i\neq 0} 2^{-c_i} \\ \Rightarrow \quad G &= \sum_{i\neq 0} c_i - \log_2 k. \end{aligned} \tag{12}$$

If we take c_i to represent the geometric deviations of each model instance from an ideal shape, then G becomes the overall geometric cost measure to within a constant.

Assuming the form (12) for the joint probability expresses the fact that parses in terms of instances with good geometric properties are more probable than others. In the absence of any geometric information, we would set $c_i = 0$, thereby effectively making a maximal entropy assumption in which one represents ignorance by assuming all prior probabilities to be equal. However, it is obvious that Eq. (12) is an oversimplification because complex scene structures typically contain mutually supporting evidence, e.g., houses on streets imply the existence of driveways. In principle one should capture these dependencies instead of ignoring them.

Thus, while the cost functions used in this work not rigorously equal to maximum likelihood measures, there is a very close relationship under the stated assumptions.

3 Implementation

Recall that our basic goal is to find parses of a scene that are optimal with respect to the objective function defined in Eq. (1) for a particular set of models.

Given unlimited computing resources, we could parse a scene in terms of a particular model vocabulary and generate labels for the image simply by evaluating the objective function for every possible combination of pixels and keeping those with the best scores. Since this is not feasible in practice, we devote this section to presenting some practical techniques for achieving our objectives without being overcome by combinatorics.

The fundamental concept is to build a set of competing and possibly conflicting model candidates using a *dynamic* procedure, and then to select the set of compatible hypotheses that maximizes the objective function. This is in fact an optimization procedure with the following components:

- State the parsing goal by defining the model vocabulary to be used in the scene description.
- Build locally optimal model primitives and retain competing candidates that exceed a threshold effectiveness score. Recurse in a hierarchy of model primitives if appropriate.
- Select a subset of compatible high-level model candidates that maximizes the objective function.

We first discuss the domains of application we have chosen, along with the specific models we have used. We then present the basic steps in the current implementation, along with some examples of how they work in practice.

Domains. We consider two principal application domains:

- **Delineation of Two-Dimensional Object Models in Aerial Imagery.** Two-dimensional feature delineation is required for the construction of maps from aerial imagery. Even when human analysts use stereographic imagery for this task, the cartographic output is always two-dimensional; three-dimensional content is indicated by iabels giving heights of building complexes, radio tower heights, and elevation contour height. Two-dimensional modeling and delineation is therefore of substantial practical importance in its own right.
- **Discovery of Three-Dimensional Object Models in Multiple Images.** Full three-dimensional data are required for applications such as flight simulation, simulation of nonoptical sensors, and autonomous ground-level navigation. The information present on typical cartographic products is no longer sufficient in these cases, and complex three-dimensional information, including perhaps even texture and material-composition maps for object surfaces, may be required. We thus consider in addition the construction of full three-dimensional models from multiple, possibly but not necessarily stereographic, images. Other modelıng information that might be used for three-dimensional reconstruction includes shadow analysis and shape-from-shading models.

Model Vocabulary. The model vocabularies that we have examined in detail include buildings, roads, and vegetation clumps in aerial imagery. The building model assumes that roof components are planes having rectilinear edges and internal areas with planar grey level intensities, with no more than 30% of the pixels being anomalous. The road model is similar, except that it assumes smoothly curved edges that are parallel and constant width apart. The tree model assumes jagged edges that have strong contrast with the background and uniformly textured interior areas. All models can in principle be optimized in stereo, so that buildings, elevated highways, and trees could be distinguished from ground-level background structures with similar geometric signatures.

Parsing Procedure. The steps in the parsing procedure that we have implemented are the following:

- **Generate Edge Cues from a Set of Segmentations.**

  Our system uses as initial shape cues segmentation region boundaries that exhibit the appropriate geometry, e.g., straight segments for buildings and elliptical segments for roads. While we could, in theory, start with the output of an edge detector, starting with the output of a region segmenter such the Laws segmenter [Laws, 1984, 1988] has proven to be very effective. The segmentation regions are optimal in the sense defined by Leclerc [1988] in terms of their area signature and are therefore good cues for the location of actual objects. Since region boundaries may not match exactly the photometric edges, the system extracts the edges by finding breakpoints in the boundary, fitting curves with the appropriate geometry between those breakpoints and optimizing their location using the technique described in Fua and Leclerc [1988]. The resulting edges are scored by computing the ratio of pixels that are maxima of the local gradient to those that are not, and only the best edges are retained.

  To achieve image independence, instead of using a single segmentation we start with a set of segmentations constructed using a family of increasingly permissive parameters, then extract edges from all the segmentation regions at all levels. This is necessary because *no single parameter setting can be expected to handle all target objects in one image, much less in multiple images.* In Figure 1, we show a typical image containing a complex suburban house; Figure 2 illustrates typical problems such as the absence of relevant shape information and the presence of excess irrelevant information encountered when one attempts to use any single parameter setting for segmentation systems or Canny edge maps.

- **Construct Compatible Binary Edge Relationships.**

  Pairs of edges that satisfy the geometric model constraints are assigned binary edge relationships. At first, in order to avoid combinatorial explosion, only edges that belong to the same segmentation regions are considered. In the next phase, the same physical edge may be identified in the boundaries of several regions belonging to different segmentations in the set; we match such edges across the set of segmentations. Using these matched edges and the already-constructed binary relationships, the system attempts to generate new relationships by considering transitive hierarchical relationships, as shown in Figure 3. Since different parts of the same object may appear at different segmentation levels, this last step turns out to be important because the system can unify shape cues that have been discovered in qualitatively different segmentations.

  Finally, a patch in the image is associated with each detected binary structure. The photometric properties of this area are computed and only structures with a good photometric score are retained.

- **Construct Cycles.** Circular lists of binary relationships satisfying the geometric model constraints are then constructed. To avoid combinatorial explosion, only binary relationships with similar photometric characteristics are used to generate cycles. In Figure 4, we show the system's symbolic representation of these cycles as circular graphs of geometric relationships. The actual edges that form these cycles are shown in Figure 5(a).

- **Build Enclosures.**

  Model-based completion procedures are now invoked to close the cycle of edges, thus generating closed contours. To complete rectilinear cycles, we use a modified version of the F^* algorithm [Fischler et al., 1981] to find paths in the lattice of most probable parallel and perpendicular edges; to complete curvilinear cycles, we use the "energy minimizing curve" technique described in Fua and Leclerc [1988]. Two such closed contours are shown in Figure 5(b). This strategy is extremely effective in finding such things as faint road edges that an edge detector may miss, but that are paired with strong edge on the opposite side of the road. A sequence of points that has been optimized as an isolated edge may move nontrivially when it is optimized as a subcomponent of a more complex model.

- **Select Enclosures.**

  Candidate enclosures are then scored using the objective functions given in Eq. (1). Enclosures with poor area characteristics or poor edge photometry are discarded. In general, for complex scenes, the system builds a set of overlapping and therefore conflicting enclosures that encompass the candidate model instances mentioned earlier. The system *chooses a subset of non-overlapping enclosures that maximizes the objective function defined in Eq. (1).*

  In applications where the system uses more than one model type, it first computes the enclosures using each model alone, then finds the best combined subset of enclosures in terms of the objective function.

4 Results

In this section, we show results of running our system in a fully automated fashion on four complex images with very different photometry. The implementation described above requires a few arbitrary thresholds such as the minimal area or edge score of an acceptable object, as well as the heuristic parameters k, p and b of Eq. (5); however, we emphasize that all results given here are obtained by running the system with a single set of thresholds and parameters. The precise form of the objective function used is

$$\begin{aligned} F &= (k-m)A + n\log_2\frac{p}{q} + \overline{n}\log_2\frac{1-p}{1-q} \\ G &= (n+\overline{n})\left(\log_2\frac{p}{q} - \log_2\frac{1-p}{1-q}\right), \end{aligned}$$

The choice of functional form for G gives preference to compact candidates by penalizing long perimeters, therefore selecting parses in which the objects are not broken into unwarranted subparts. The choice of coefficients ensures that high-quality edges are penalized less than poor quality ones. We fix the values of the parameters to be $k = 7.0, p = 0.9, q = 0.01$, and impose the thresholds $m < 7.0$, $n/(n+\overline{n}) > 0.7$. Parsing results are relatively insensitive to changes in the parameters. The value 7.0 is interpretable as the number of uncorrelated bits in one of our typical digital gray-scale images; 0.9 is approximately the probability that a boundary pixel of an object is an optimal edge and 0.01 is the approximately the ratio of optimal edge pixels to non-optimal pixels in the images we have been working with.

4.1 Buildings and Roads: Rectilinear and Curvilinear Networks

To illustrate the behavior of the system on monoscopic data, we have chosen two images that are especially challenging because of shape complexity and faint edge photometry. When the first image, Figure 1, is analyzed using the building and the road models separately, we find that the highest-scoring set of non-overlapping enclosures consists of those shown in Figures 6(a) and (b). Merging the two models, we find the overall parse in terms of the best candidates for buildings and roads shown in Figure 6(c). We see that conflicts are resolved in favor of the model with the strongest evidence. Some false model candidates remain because, in the absence of stereoscopic or shadow information, rectilinear yard areas are difficult to distinguish from legitimate buildings. In this particular case, they can be eliminated by retaining only the enclosures that have the highest score, as shown in Figure 6(d). The incorporation of stereo into the analysis is discussed below.

The next image, Figure 7, also contains difficult-to-parse cultural structures with a different physical scale. Figures 8(a) and 8(b) show the building and road candidates found by applying the models separately; all of the apparent buildings but one were found. When the two models are merged, we find the results in Figure 8(c); by retaining only the best enclosures, we get the final parse shown in Figure 8(d).

4.2 A Group Picture—Houses, Roads, and Trees

Vegetation clumps, typically small groups of trees, are complementary to the regular cultural-object models we have described so far. Their boundaries are typically jagged and irregular. Any compact object that contrasts strongly with its background context and has no components that are road-like or building-like could be a candidate for vegetation.

We generate candidate vegetation clumps by optimizing the locations of segmentation region boundaries. To compensate for the lack of geometric constraints, we use two additional thresholds for scenes requiring vegetation discovery: For any candidate instance, we require a minimum acceptable compactness, as measured by (A/P^2), where A is the area and P the perimeter, and a minimum contrast, as measured by the ratio of the average edge strength along the boundary to the average edge strength inside the region.

We now examine the monoscopic image in Figure 9 and parse it with respect to all three models buildings, roads, and trees. The separate results for each model are shown in Figure 9(a) and (b); the main road is lost because of its low contrast.

The tree model can be extended to the stereographic case by requiring jaggedness in the relative elevation of successive points on the candidate vegetation curve; this characteristic distinguishes vegetation from shadows of vegetation and other irregular ground markings.

4.3 Stereographic Buildings

When stereographic or multiple imagery is available, we generalize the model to account for elevation of the rectilinear components above the ground or above enclosing building components. Both horizontal and slanted roof planes are permissible, and the three-dimensional generic model consists of planar surfaces, each of which has at least one straight edge in common with another surface. The additional term used to evaluate stereo match quality in the objective function is

$$F_{\text{stereo}} = (7.0 - m_s)A, \tag{13}$$

where m_s was introduced in Eq. (6).

In Figure 10, we show a stereo pair containing a complex set of buildings. Running the monoscopic procedure on one image, we get the rectilinear outlines shown in Figure 11. Next, we use the camera models to back-project these outlines into the second image and use the stereographic-matching objective function to optimize the height of the building sections. We thus obtain the three-dimensional structures that exhibit the best agreement with the model constraints and the evidence in the stereographic image data. In Figure 12, we selectively display the structures that have significant elevation with respect to the ground.

Since we know the three-dimensional position of each rectilinear contour, we can now generate three-dimensional objects having the observed two-dimensional upper surface. When we supply the automatically computed surface contours of the large building in Figure 10 to the SRI cartographic modeling system [Hanson et al., 1987; Hanson and Quam, 1988], we can generate synthetic three dimensional views of the scene such as the one shown in Figure 13.

5 Conclusions

In this work, we have proposed a scene-modeling approach based on a hypothesis-generation technique that uses active optimization to discover generic model components in digital images.

The objective function that is optimized measures the closeness of each model hypothesis to an ideal model instance and to the evidence supporting its realization in the image data.

We apply this objective function in practice by generating model candidates from segmentation-based cues and by optimizing the geometry of the model candidates with respect to the measure. The model-instance discovery process optimizes the model interpretation by referring directly to the image data at all levels of the scene parsing process.

We have used generic geometric models that include a combination of edge, photometric, and stereographic characteristics to delineate several classes of objects in aerial images. Invocation of a particular model or set of models effectively sets a goal for the image interpretation process, giving us the freedom to parse the same image with different objectives.

The system's effectiveness derives from the use of the objective function philosophy to optimize the entire interpretation process with respect to the evidence in the image. In future work, we plan to extend the range and complexity of the models supported, to integrate geometric measures more closely with the photometric effectiveness in our overall framework, and to include high-level interdependencies among model instances to support complex cartographic analysis requirements.

References

S. Barnard, "Stochastic, Hierarchical Stereo Matching," in this Proceedings.

T.O. Binford, "Survey of Model-Based Image Analysis Systems," The International Journal of Robotics Research **1**, No. 1, pp. 18-64 (Spring, 1982).

R.C. Bolles and R. Horaud, "3DPO, A Three-Dimensional Part Orientation System," International Journal of Robotics Research **5**, pp. 3-26 (1986).

R.A. Brooks, "Symbolic Reasoning Among 3-D Models and 2-D Images," Artificial Intelligence Journal **16**, (1981).

J. Canny, "A Computational Approach to Edge Detection," IEEE Trans. PAMI **8**, pp. 679-698 (1986).

P. Cheeseman, "A Method of Computing Generalized Bayesian Probability Values for Expert Systems," in Proceedings of the Eight International Joint Conference on Artificial Intelligence; IJCAI-83, pp. 198-202 (1983).

M.A. Fischler, J.M. Tenenbaum, and H.C. Wolf, "Detection of Roads and Linear Structures in Low-Resolution Aerial Imagery Using a Multisource Knowledge Integration Technique," Computer Graphics and Image Processing **15**, pp. 201-223 (1981).

P. Fua and A.J. Hanson, "Locating Cultural Regions in Aerial Imagery Using Geometric Cues," Proceedings of the Image Understanding Workshop, Miami Beach, Florida, pp. 271-278 (December 1985).

P. Fua and A.J. Hanson, "Resegmentation Using Generic Shape: Locating General Cultural Objects," Pattern Recognition Letters **5**, pp. 243-252 (1987a).

P. Fua and A.J. Hanson, "Using Generic Geometric Models for Intelligent Shape Extraction," Proceedings of the Image Understanding Workshop, Los Angeles, California, pp. 227-233 (February 1987b).

P. Fua and A.J. Hanson, "Using Generic Geometric Models for Intelligent Shape Extraction," Proceedings of the AAAI Sixth National Conference on Artificial Intelligence, pp. 706-711 (July 1987c).

P. Fua and Y.G. Leclerc, "Model-Driven Edge Detection," in this Proceedings.

A.J. Hanson, A.P. Pentland, and L.H. Quam, "Design of a Prototype Interactive Cartographic Display and Analysis Environment," Proceedings of the Image Understanding Workshop, pp. 475-482 (February 1987).

A.J. Hanson and L.H. Quam, "Overview of the SRI Cartographic Modeling Environment," in this Proceedings.

R.M. Haralick, "Digital Step Edges from Zero Crossings of Second Directional Derivatives," IEEE Trans. PAMI **6**, pp. 58-68 (1984).

K.I. Laws, "Goal-Directed Texture Segmentation," Technical Note 334, Artificial Intelligence Center, SRI International, Menlo Park, California (September 1984).

K.I. Laws, "Integrated Split/Merge Image Segmentation," Technical Note, Artificial Intelligence Center, SRI International, Menlo Park, California (February 1988).

Y.G. Leclerc, "Image Partitioning as Constructing Simple Stable Descriptions," in this Proceedings.

Y.G. Leclerc and P. Fua, "Finding Object Boundaries Using Guided Gradient Ascent," Proceedings of the the 1987 Topical Meeting on Machine Vision, Lake Tahoe, California, pp. 168-171 (March 1987). Also presented at the DARPA Image Understanding Workshop, Los Angeles, California, pp. 888-891 (February 1987).

J. Rissanen, "Minimum-Description-Length Principle," in *Encyclopedia of Statistical Sciences*, **5**, pp. 523-527, (1987).

M.O. Shneier, R. Lumia, and E.W. Kent, "Model-Based Strategies for High-Level Robot Vision," Computer Vision, Graphics and Image Processing **33**, pp. 293-306 (1986).

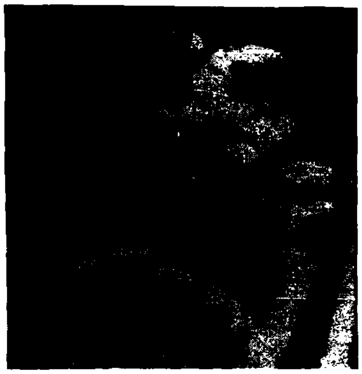
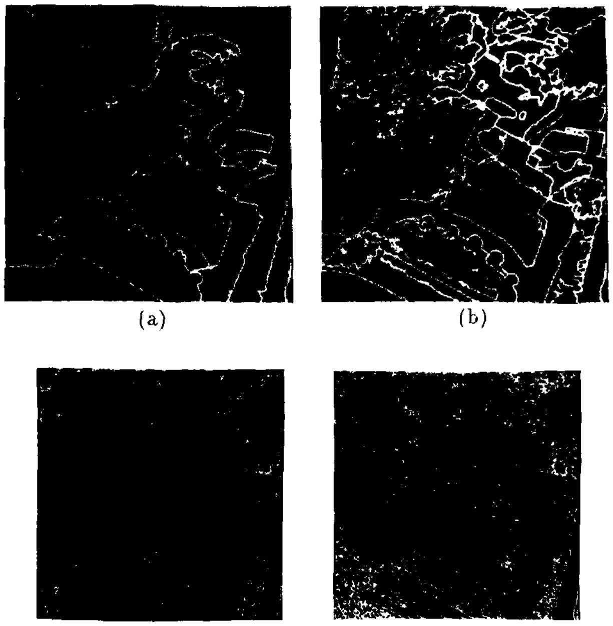


Figure 1: A complex house in a suburban context.



(a) (b)

(c) (d)

Figure 2: (a) A Laws segmentation with an undersegmented roof. (b) Oversegmentation resulting from different parameter choice. (c) A Canny edge map showing a scarcity of object edges. (d) Excessive irrelevant edges resulting from a different edge threshold choice.

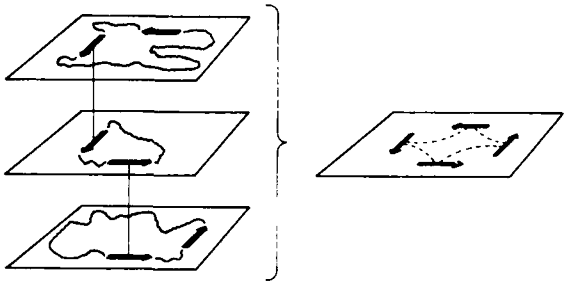


Figure 3: Construction of binary relationships among elementary edge structures across a set of segmentations.

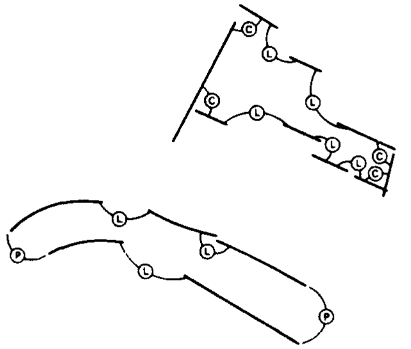


Figure 4: The symbolic relationships among the edge components of two typical cycles. *L*, *C* and *P* stand for linear, corner, and parallel binary relationships, respectively.

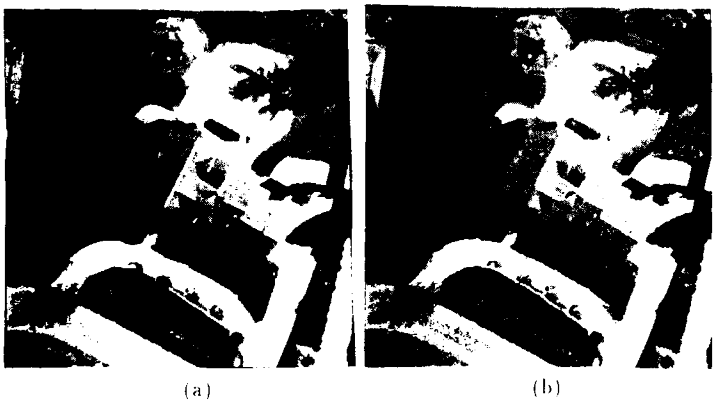


Figure 5: (a) A rectilinear cycle and a curvilinear cycle of edge segments overlaid on the image. (b) The results of running the model-driven contour completion algorithm on each of the cycles of binary relationships to generate complete closed contours.

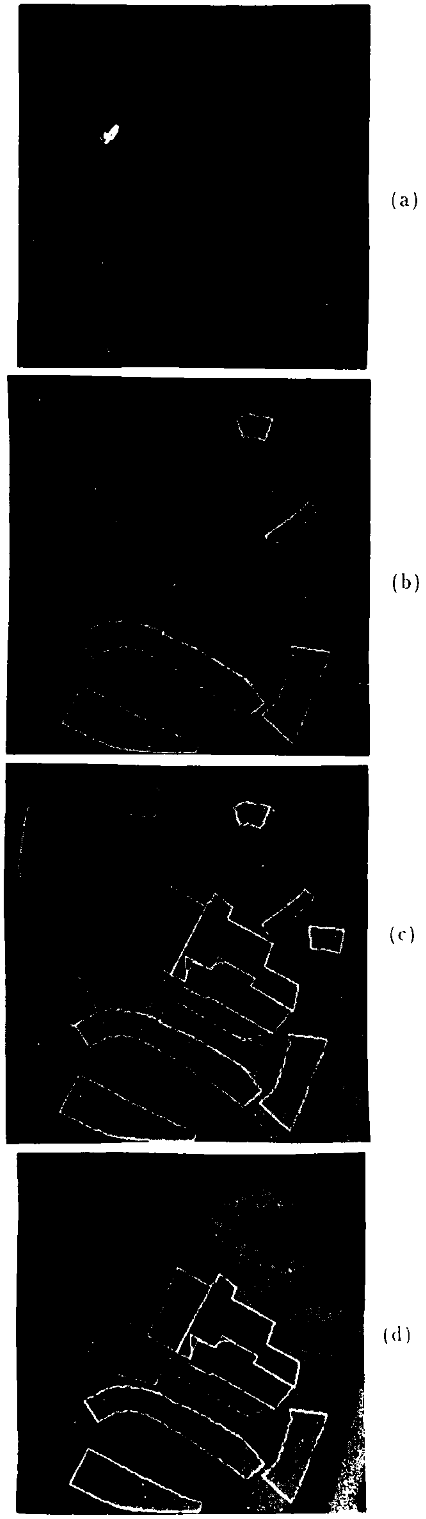


Figure 6: (a) Candidate building structures. (b) Candidate road structures. (c) Merged building and road structures. (d) Final parse in terms of the best structures.

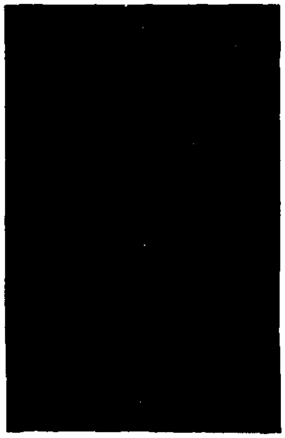
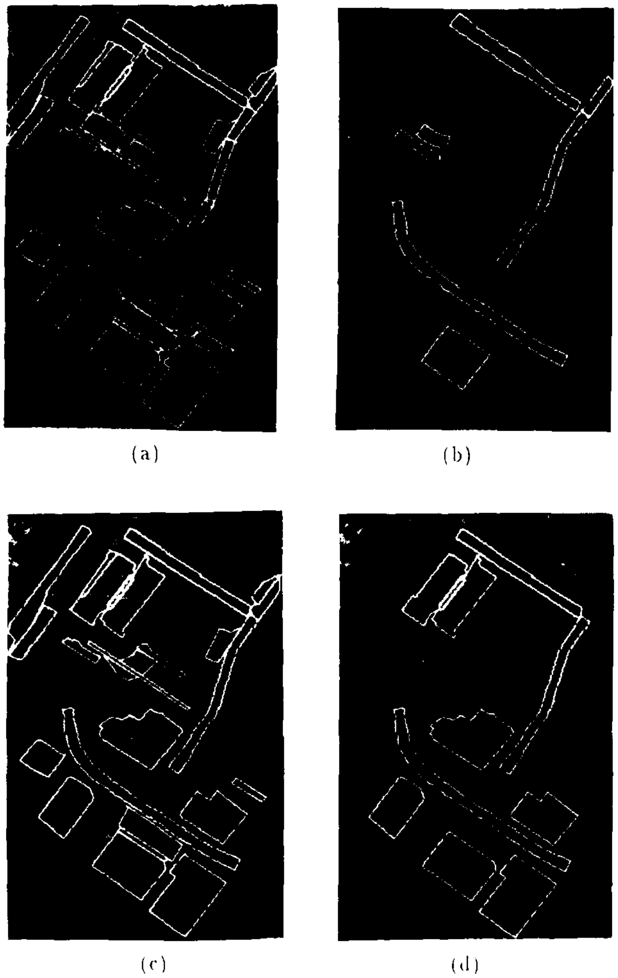


Figure 7: An aerial scene with industrial buildings and roads.



(a)

(b)

(c)

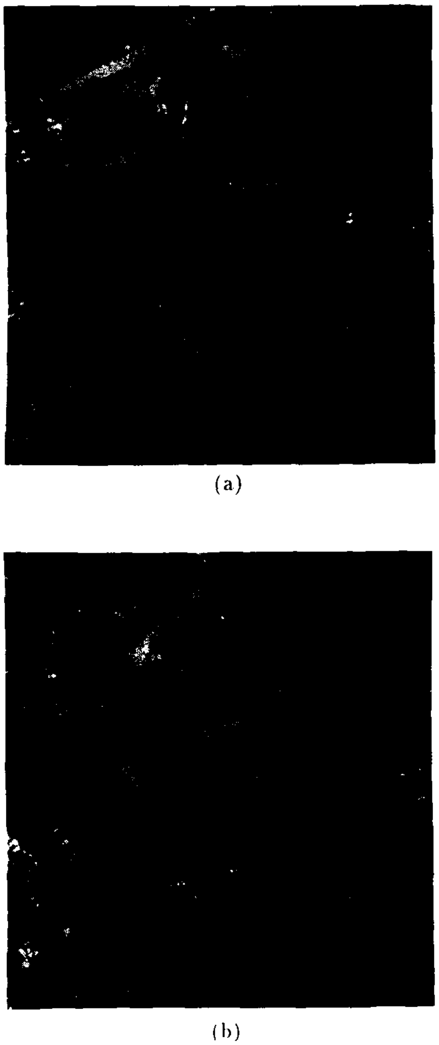
(d)

Figure 8: (a) Candidate building structures. (b) Candidate road structures. (c) Merged building and road structures. (d) Final parse in terms of the best structures.

(a)

(b)

Figure 9: (a) Tree candidates. (b) Building candidate.



(a)

(b)

Figure 10: A stereo pair of images containing complex large buildings.

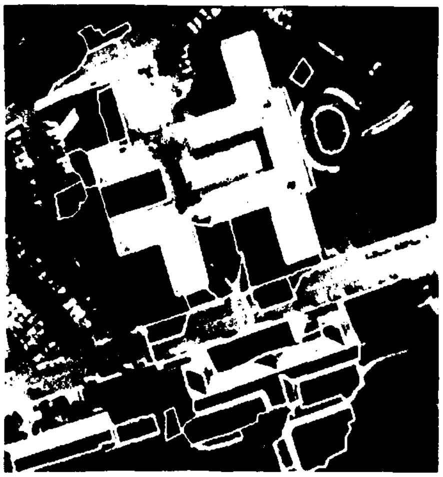


Figure 11: All candidate rectilinear structures.

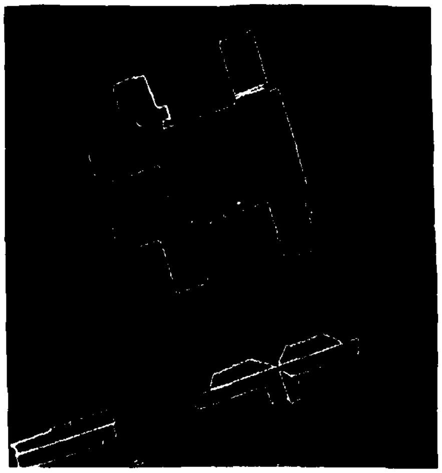
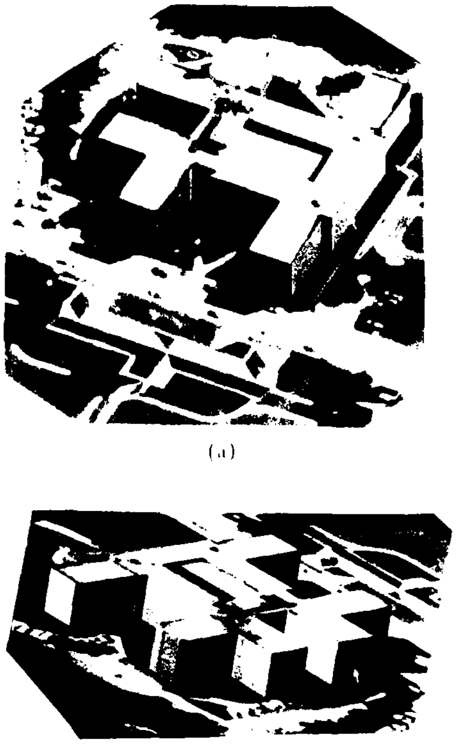


Figure 12: The best parse of the scene into components with substantial elevation in three dimensions.



(a)

(b)

Figure 13: Synthetic views of a three dimensional scene with the original image texture-mapped onto the central building model. Scene areas that cannot be texture-mapped because they are not visible from the original camera position are shown in black.

Texture Models and Image Measures for Segmentation

Richard Vistnes

Robotics Laboratory, Stanford University
Stanford, California 94305

Abstract

The problem of texture segmentation is to identify image curves that separate different textures. The segmentation problem has two phases: detecting differences between textures, followed by localizing the edges thus detected. This paper focuses on texture discrimination, which is based on differences in texture measures between two image regions. We treat texture not merely as image pattern, but as the projection of scene structure; two image textures differ when the scene textures of which they are projections differ.

A model is proposed for a large class of scene textures and examines their images under projection. A set of image texture measures is proposed that allows the reliable discrimination of physically different textures. Textures are characterized by distributions of shape, position and color of image substructures; methods for estimating these distributions are discussed. A forced-choice method is proposed for evaluating the ability of texture measures to discriminate textures. The proposed texture measures have been implemented, and empirical tests show that the measures can discriminate a large set of natural textures with 90–100% accuracy, compared with about 40% accuracy for another popular set of texture measures.

1 Introduction

Texture segmentation seeks to identify image curves on either side of which textures differ. Such image texture discontinuities usually correspond to some kind of discontinuity in the scene[1], such as an object boundary, occlusion, material change, etc. A texture segmentation algorithm bases its edge decisions on the results of a set of *texture measures* applied to the image; it detects edges where the measures change abruptly. One goal of texture segmentation research is to find a set of texture measures that serves to discriminate between a large proportion of different textures. Many such texture measures have been proposed in the literature, some ad hoc, some motivated by human psychophysical data. Most of these have treated image texture as simply pattern in the image, failing to recognize that we seek to discriminate textures that differ in the *scene* rather than in the image. To this end, it is necessary to model texture in the scene, examine its projection to the image, and measure image features that serve to discriminate different scene textures. This paper proposes such a texture model, derives the associated image features, and demonstrates their utility in distinguishing real textures.

To begin with, it is necessary to summarize the main points of an earlier paper [27]. The remainder of this introduction discusses the notion of a texture generation process as a way to model texture, and examines the meaning of texture segmentation. Section 2 proposes a method for quantifying the ability of a set of image texture measures to discriminate textures. Section 3 examines the nature of textures in the 3D scene and proposes a model. The following section examines the image of such textures under projection, and derives image measures that serve to distinguish different scene textures. Section 5 presents results that show that the proposed image measures can reliably distinguish a variety of natural textures, and compares these results with those obtained with other texture measures. The final section recapitulates and identifies directions for further research.

1.1 Texture Processes and Measures

The problem of texture segmentation may be broken into two phases: detecting differences between image textures, and localizing the texture edges thus detected. This paper focuses on the former aspect, texture discrimination; the goal is to determine whether two image textures correspond to different scene textures. Textures may differ in two ways: physically and perceptually. Consider an analogy to color perception[2]: two colors are *physically* identical when their color spectrum is identical, and *perceptually* identical when they cannot be distinguished by a human observer. The colors we distinguish are determined by our set of color receptors; physically different colors that stimulate the color receptors identically are perceived as identical, and are called *metamers*.

To make the analogy to texture, we must be able to describe a texture physically. For this purpose, we use the parameters of the process that generated the texture. A *texture generation process* is the physical source from which

[1] We use the convention that *scene* refers to a three-dimensional collection of objects, of which the *image* is the two-dimensional projection.

[2] This analogy will be useful at other points in this paper.

a scene texture arises; the process is governed by a set of parameters that determine the nature of the textures created. We distinguish between a texture generation process and its *instances* in the scene (which are three-dimensional), and the images of such texture instances. Texture generation processes are often stochastic. As an example, a process that generates a three-dimensional collection of randomly-placed dots is a texture process (its parameters include the dot density and placement distribution) and any particular collection of such dots is an instance of the texture. In this paper, the term *texture* refers to texture processes and texture instances in the scene, i.e., three-dimensional textures, while *image texture* refers to the 2D projection of an instance of a scene texture. We will often refer to an instance of a scene texture simply as a scene texture, when there can be no confusion.

To continue the analogy with color, two texture processes are physically identical when their parameters are identical. Two image textures are physically identical when they are the images of identical scene textures (assuming identical imaging geometry). Two image textures are perceptually identical when a human observer cannot preattentively discriminate them; these depend on the texture features measured by our visual system. Physically different textures may be perceptually indistinguishable (metamers). For example, the textures in Fig. 1a differ physically and perceptually, while those in Fig. 1b are metamers, since human vision measures no texture features that allow immediate discrimination. The textures that a vision system can distinguish depend on the texture features it measures. A vision system that measures a finite set of features cannot distinguish every possible pair of physically different textures just as a color vision system that measures a finite number of color samples cannot distinguish between all physically different colors [4]. However, carefully chosen color sensors allow discrimination of most physically different colors [13]; similarly, carefully chosen texture measures allow us to distinguish most physically different textures as well. This paper focuses on that choice of texture measures.

1.2 Texture Edge Detection

When texture processes are stochastic (as is usually the case) it is not possible in general to decide without error if two textures are different or not. As a consequence, there is no unique way to segment a textured image. A simple thought experiment should clarify this: imagine an image in which each pixel is chosen randomly to be white or black, with equal probability. Such an image contains no underlying edge, and on average looks 'random.' However, it is possible (albeit unlikely) that all the pixels on one half of the image are black while all of those on the other half are white. It is incorrect to infer that a discontinuity exists in the underlying distribution. Instead, one may compute the statistical significance of the observed edge, that is, the probability of the observed difference, under the null hypothesis that the textures are identical. Such tests do not lead to binary decisions (edge, no edge) unless a significance level is chosen. In the random-pixel example, the significance of an edge between all-black and all-white regions would be very high, and our confidence in an edge at that location is high. In general, we can at most estimate relevant image parameters on either side of the edge and compare them; our certainty that there is an edge depends on the size of the difference in estimated parameters. When the estimated edge probability is low, the observed edge is likely to be the result of a true scene discontinuity.

This notion of edge detection leads to testing for edges at many positions, orientations, and scales in the image. As a measure of the difference between two image textures, we shall use the statistical significance of the difference between estimated distributions. (Note that one determiner of the significance of the edge is the size, or scale, of the region over which statistics are collected.) This difference measure will be used in the localization stage to isolate edges at places where the difference is locally maximum. Statistical significance gives a precise meaning to the 'output of the edge detector' and provides a solid foundation for further interpretation.

2 Evaluating Texture Measures

There are no commonly accepted criteria for judging the effectiveness of a texture segmentation algorithm; this lack makes it difficult to compare texture measures with respect to their ability to distinguish textures. Researchers typically demonstrate a segmentation program by showing that it finds the major texture edges in a natural image [18,19,23,30], or by constructing a mosaic of textures and showing that the algorithm correctly segments the mosaic [5,6,20]. These tests confound the ability of a set of texture measures to *discriminate* textures with the ability of the algorithm to *localize* edges. This paper is concerned primarily with texture discrimination; again, one goal of this research is to determine a set of texture measures that serves to discriminate most physically different textures. This section discusses ways to evaluate texture measures. The texture model and associated image features proposed here are later evaluated according to this method.

Suppose we are to evaluate a set of image texture measures M with respect to its ability to discriminate textures. Let $\mathcal{T}$ be a texture process and T be the image of an instance of $\mathcal{T}$. Denote by $M(T)$ the result of applying the measures M to T; $M(T)$ is a vector (possibly of length 1) that serves to describe T. We will adopt what is known in psychology as the *forced-choice paradigm* for evaluating texture measures. The system is presented with images in which it is known that there is a texture edge at one of N positions; it must determine where the edge is located. The location decision corresponds to the position where the maximum difference between texture measures occurs. The fraction correct is accumulated over a number of trials. When M cannot discriminate $\mathcal{T}_1$ and $\mathcal{T}_2$, performance is near the chance level $1/N$. This evaluation method has the advantage that it does not use thresholds, nor does it require edges to be localized. For use below, let us denote the performance (fraction correct) of M on $\mathcal{T}_1$ and $\mathcal{T}_2$ by $P_M(\mathcal{T}_1, \mathcal{T}_2)$.

It is useful to quantify the performance of a set of mea-

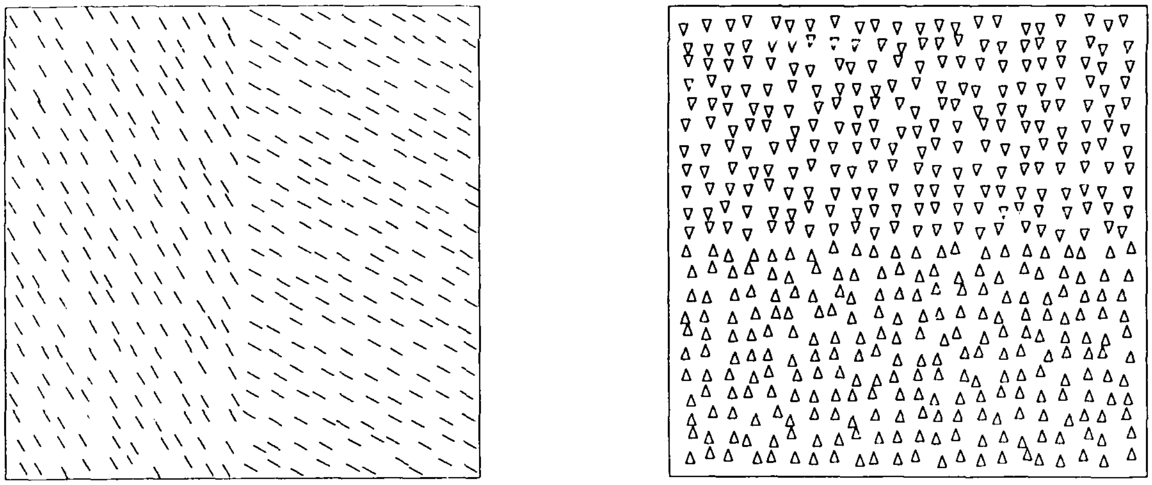


Figure 1: Examples of preattentively distinguishable and indistinguishable textures. (a) Difference in orientation of line segments causes segmentation. (b) No texture segmentation, although upon inspection the triangles on top and bottom are inverted. This physical texture difference is not immediately perceived.

sures M over a population of textures that are likely to be encountered. Suppose we restrict our attention to such a class C of textures. Each member of C is a texture generation process defined by some set of parameters that controls the process. Describing the distribution of textures in C requires describing the distribution of texture parameters. This distribution may be very complex and difficult to describe, but it does exist. We measure the performance of M on C by repeatedly choosing two texture processes $\mathcal{T}_1$ and $\mathcal{T}_2$ from C, according to the distribution, and determining the performance $P_M(\mathcal{T}_1, \mathcal{T}_2)$ of M. The average performance measure provides an indication of the ability of M to distinguish two textures from the class C. When C is the class of textures likely to be encountered in the natural environment, the distribution of textures in C is very complex, and it is necessary to determine the performance of M empirically, that is, by testing it on examples of textures. One way to do this is to construct test images from textures drawn from a large sample such as Brodatz's album [3]. Since the texture process itself is not available to generate textures, we must be satisfied with using different portions of a single instance. This is the approach taken here.

3 A Model for Scene Texture

We now begin the discussion of models for scene texture. Previous models for texture have been confined primarily to models of image texture (e.g., [1,8,11,17]). These image models are of limited interest to us here. Shape-from-texture methods usually make the model for scene texture more explicit, since they must acknowledge the fact that image texture is the projection of scene texture. These models usually assume that textured objects have smooth surfaces covered with uniform-density, identical texture elements [2]; elongated texture elements are assumed to be isotropically distributed [16,31]. Such models are unrealistic in natural scenes; here we discuss a more plausible model for scene texture. This paper provides only a brief description of the model; more details may be found in [28].

3.1 Examples

The scope of the scene texture model includes the following kinds of objects whose images are textured; this list is meant to be suggestive, not exhaustive.

- Homogeneous, connected objects that appear identical everywhere, such as a volume of air, water or metal. These are instances of null textures, discussed below.
- Collections of physically separate and well-defined parts, normally held together by gravity, such as a pile of rocks, sand, etc.
- Collections of physically separate parts distributed in a background medium; the background is in general another texture. Examples: a cluster of balloons, a flock of birds, rocks in snow.
- Variegated, connected objects, such as wood, granite, marble, dirt, etc. Such an object can be viewed as a collection of substructures whose boundaries are poorly defined; its surface may be rough or smooth.

In addition, we allow for texture arising from object surfaces, including:

- Surface-marking texture, i.e., zero-thickness markings on a smooth surface. This is the class of scene textures to which most previous models (e.g., [2,16]) have been restricted. Such surfaces have been modeled by mapping a texture to a curved surface, or deforming a planar texture [7].

- Surface-structure texture. This class encompasses surface roughness on snow and dirt, bricks, etc. The surface and the interior are composed of the same matenal.
- Surface-covering texture. This class includes grass, carpet, etc., where the surface and underlying material differ.

The image textures formed by projection of these classes of scene textures are very diverse and very complex. Because of this complexity, we cannot model them in detail; fortunately, for segmentation purposes there is no need to model them in detail. We need only measure image features that allow us to discriminate reliably between physically different textures.

3.2 The model

We model scene textures by *scene texture processes*, which fill a region of space with a null texture and a possibly empty collection of substructures, each of which is again described by a scene texture. The *null texture* is the simplest scene texture; instances of null textures are everywhere identical at the finest resolution. An *instance* of a scene texture is the three-dimensional region filled by the process.

The substructures in a non-null texture are chosen from some statistical distribution, and placed (in orientation and position) according to another distribution. We characterize a substructure by its shape, its position (including orientation), and its subtexture (including color). A non-null texture process is specified by distributions of shape, position, and subtexture of its substructures, and by the color of its null texture background. For example, a sand texture is composed of grains of sand; each grain is chosen from a distribution that specifies its size, shape and color, and it is positioned and oriented according to another distribution.

We model texture arising from object surfaces in a similar manner. Surface-structure texture and surface-covering texture are modeled as smooth surfaces overlaid with substructures to provide the surface texture. Such substructures are again chosen from statistical distributions that specify their shape, subiexture, and position and orientation on the surface. Similarly, surface-marking textures may be modeled as smooth surfaces covered with two-dimensional substructures chosen from distributions specifying their attributes and positions.

4 Image Features

For texture discrimination, we wish to determine a set of image texture measures that is sufficient to discriminate a large number of physically different textures. To do this, we examine the appearance in the image of scene textures conforming to the model proposed above, and derive texture features that serve to distinguish textures with different parameters. This paper provides only a brief summary of the derivation of the texture measures; more details may be found in [28].

To discriminate null textures, measurement of image color generally suffices; differences in image color usually imply differences in object color (assuming common illumination). Regions differing in color spectra differ as well in intensity, except on a set of measure zerc, and hence intensity cues suffice to detect almost all color edges (see [12]). We detect edges in null textures where image intensity changes abruptly; this corresponds to the classical notion of edge detection.

To discriminate non-null textures, we wish to compare distributions of shape, position, and color of image substructures. We base comparison of substructure shape on comparison of overall length and width of image substructures, and on length of image edges. We also estimate distributions of overall orientation of image substructures and image edges. The substructures composing image texture are generally difficult to isolate (e.g., in grass or carpet), so it is difficult to measure their attributes directly. Instead, we use *feature detectors* that are tuned to image structures with particular attributes. For example, we use an elongated center-surround detector to detect image regions of certain length and width that differ in mean intensity from the local surround. Summing the responses of these detectors in an appropriate manner provides an estimated histogram for a single feature, as follows.

Suppose that image structures could be isolated and we could measure their attributes (e.g., length, orientation) directly. We could estimate histograms of these attributes by dividing the domain of each attribute into a number of ranges (bins), and counting the number of structures whose attribute value falls into each range. We may compute approximations to such histograms by applying feature detectors to the image. A feature detector is tuned to image structures with a particular set of attributes, such as size, elongation, orientation and color. Each feature detector computes a measure of the confidence in the presence of a structure at the detector's location, with value 1 indicating high confidence that there is a structure whose attributes match those to which the detector is tuned, and 0 indicating low confidence. Suppose again that image structures are well defined, and suppose further that feature detectors respond with 1 only when situated on an image structure whose attributes match those to which the detector is tuned, and 0 otherwise. Then we may construct a histogram for, say, orientation, by evaluating feature detectors at densely sampled locations within a region, and summing responses for each orientation bin over all sizes, elongations and colors. The resulting histogram will be identical to the one computed by isolating the structures and measuring their features. The advantage of this procedure is that it can be applied to images in which structures are not well defined.

To estimate distributions of overall structure shape and orientation, we employ elongated structure detectors (ESD's) that test for image structures by comparing average color inside and outside an elongated region. Fig. 2 shows such a detector; its parameters are given by length L, elongation $E = w_c/L$, orientation θ, and a color sensitivity function $C(\lambda)$ that specifies the spectral response of the detector (cf. [13]). Image substructures may be elongated or not; they appear in all sizes and elongations. Since the

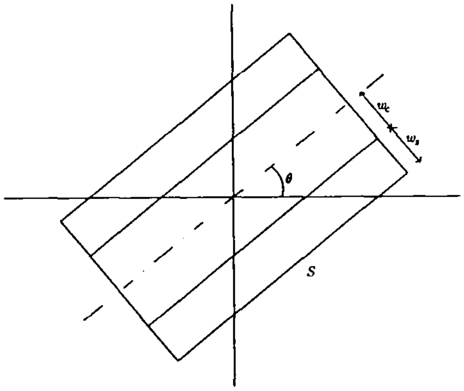


Figure 2: An elongated structure detector. It computes the significance of the difference between pixel means in center and surround.

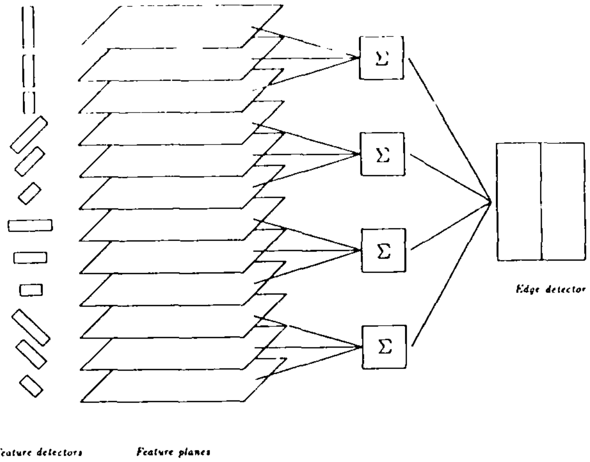


Figure 3: Illustration of feature summation for edges in ESD orientation. Responses of ESDs of various lengths and elongations are summed in each slice at each orientation to form a histogram vector (here, with four components).

size of image substructures is unknown in advance, we must employ a variety of lengths L of detectors. Corresponding to each length are a variety of elongations E ranging from 1 (circular or isotropic) to $E \ll 1$ (narrow). For each of these length/elongation pairs we employ a variety of orientations θ. For each detector in this three-dimensional collection we employ a set of n spectral sensitivity functions; in typical situations, $n = 3$ for color images (e.g., red, green, blue) and $n = 1$ for black and white images. For simplicity in what follows, we shall restrict attention to black and white images.

Each ESD computes the statistical significance of the difference D between the means of pixel gray levels in its center region and in its surround region, using a simple t-test [24]. The operator computes $1 - P(D \mid H_0)$, where H_0 is the null hypothesis that the distributions are identical. This response approaches 1 as the structure becomes more significant. Let us denote the response of an operator of length L, elongation E, orientation θ, at image location (x, y), by $R(L, E, \theta, x, y)$. We wish to estimate histograms of L, E and θ of structures. Suppose we divide the L domain into M_L bins, $L_1, L_2, \ldots, L_{M_L}$; similarly, we divide E into $E_1, E_2, \ldots, E_{M_E}$ and θ into $\theta_1, \theta_2, \ldots, \theta_{M_\theta}$. The histogram for L is a vector with M_L components, $(l_1, l_2, \ldots, l_{M_L})$. Each component l_i is the sum, over the image region under consideration, of $R(L_i, E_j, \theta_k, x, y)$, where $1 \leq j \leq M_E$ and $1 \leq k \leq M_\theta$, and x and y range over the region. We compute the histograms for E and θ in an analogous manner. Figure 3 illustrates this feature-value summation for the θ feature.

To determine the significance of any difference between two histograms, we use the edge detection method proposed in [27]. Briefly, an edge detector such as the one pictured in Fig. 4 is used. It divides its receptive field on either side of a hypothesized edge into N 'slices' and computes an averaged histogram vector for each slice. Multivariate regression techniques are used to fit lines to the data on either side of the edge, and to compute the intercept vectors at the center along with their associated covariance matrices. Edges are manifested as differences in the intercepts; the operator calculates the statistical significance of any difference. The image of an object with a curved surface exhibits smoothly changing texture features; this edge detector is insensitive to such smooth variation. (This type of edge detector has its roots in the statistical techniques for intensity edge detection proposed by Haralick [10,9] and Leclerc [21].)

To estimate distributions of length and orientation of the edges that form substructure boundaries, we use intensity edge operators of various lengths and widths. These operators are like the ones just described (see Fig. 4) except that they average image intensity over each slice. To obtain orientation and length histograms, we employ intensity edge operators of several lengths and orientations, at densely sampled positions in the image. The responses are summed across length and orientation, as for the ESDs, to calculate the histograms. We also use this type of intensity edge operator to compute the significance of any overall difference in intensity or color between two texture regions. The overall color difference may be used for discrimination when there are no differences in other texture features; this allows segmentation using intensity, as in classical edge detection.

4.1 Feature combination

We must combine the significances of differences between individual distributions to obtain the overall significance of the texture difference between two regions. It simplifies matters greatly if we assume that the distributions are independent, i.e., that the attributes (such as shape and color) of a substructure are chosen independently. In addition, we assume that the length and orientation of substructure

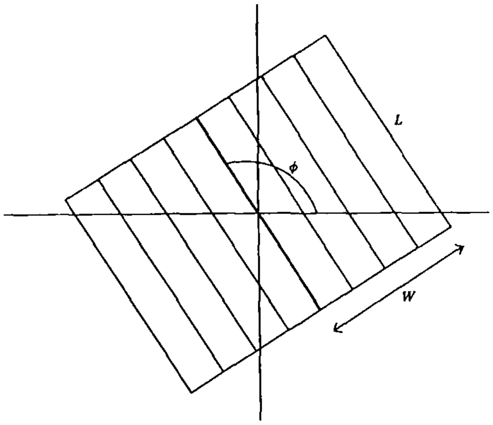


Figure 4: An edge detector. It divides its receptive field into N slices (here, $N = 4$) on either side of a hypothesized edge, as shown.

edges are independent from the overall size and orientation of substructures. While the latter assumption is clearly an oversimplification, it is a reasonable approximation, particularly since the small-scale edges bounding a structure are often created by processes independent from those which create the structure as a whole. Similarly, we may treat any overall difference in intensity or color between two regions as independent from differences in distributions of substructure features, although this is again an oversimplification.

Under the assumption of independence, the probability of observing all feature differences between two regions simultaneously is the product of the individual probabilities, so the overall significance of the texture difference is just the product of the individual significances. When feature distributions are in fact correlated, the computed overall significance overstates the true significance of the edge. It is worth noting that human vision appears to assume independence of features as well, since we cannot perceive texture differences defined by a conjunction of features [25,26,14].

5 Empirical Results

The acid test of a generally useful set of texture measures is its ability to distinguish many different textures. Proof of such an ability rests mainly upon empirical tests, which are the subject of this section. A forced-choice test such as that discussed in Section 2 was conducted on a number of texture pairs constructed from images taken from Brodatz's [3] album. A 4-alternative test was used; the possible edge locations were vertical at left and right, horizontal at top and bottom, placed symmetrically. (For simplicity, the edge was always placed at the bottom, though this was not known to the edge detection program.) Figures 5 and 6 show two of the texture pairs tested. Although an accurate test would use many texture trials to determine $P_M(T_1, T_2)$ for a texture pair, these tests used only one trial; these results should therefore be viewed as an approximation. In these tests, no preprocessing was performed on the image to equalize histograms or overall intensity.

This section reports the results of three series of experiments. These experiments determined the discrimination ability of three different sets of texture measures on a set of 45 texture pairs. The first and second series used the same set of texture pairs, while the third used a different set of textures. We discuss the first series in detail, then summarize the results of the second and third series.

In all experiments, distributions of five features were estimated: size, elongation and orientation of elongated structures, along with length and orientation of intensity edges; averaged intensity was used as the sixth feature. The images were 256 × 256 pixels. In the first series, the ESD's were of two sizes S (8 and 12 pixels); they each had two elongations E (0.7 and 0.30); and they had four orientations θ (0, 45, 90 and 135 degrees). They used equal center and surround areas, so $w_c = w_s$ (see Fig. 2). Elongation of 1.0 was treated as an isotropic operator; these were circular center-surround operators with no orientation and diameter equal to S. There were thus $2 \times 2 \times 4 = 16$ different ESD's and two sizes of isotropic operator. There were two lengths L of intensity edge detector, again 8 and 12 pixels, and the same four orientations ϕ. The width of the small edge detectors was a constant fraction of the length, $W = 0.6L$, and they used $N = 4$ slices.

To test for differences in individual distributions, the feature edge detectors discussed in Section 4 (see Fig. 4) were used. Each of these edge detectors had length $L = 80$ pixels, width $W = 50$, and used $N = 5$ slices, except in the case of orientation which used $N = 6$. The average-intensity edge detector had the same dimensions and $N = 4$. For ease of interpretation, the operators in the implementation computed the negative logarithm of the significance; this value grows with the edge significance.

Discrimination tests were carried out for the 45 texture pairs constructed from 10 texture images taken from Brodatz. Figures 5 and 6 show two of the images tested; Tables 1 and 2 contain the negative log significance of the texture edges at four locations. The six features are: size S, orientation θ and elongation E of elongated structures, length L and angle ϕ of intensity edges, and averaged intensity I. The overall edge significance is the sum of the individual significances. Table 3 summarizes all the results. Each entry in this table contains two symbols: the first indicates results using all features, including the overall intensity difference, the second indicates results excluding intensity. In this series of tests, the textures were correctly discriminated in 41 images, or 91.1%. When the overall intensity difference was not used for discrimination, accuracy was still high, 38 of 45, or 84.4%.

It is of interest to see how discrimination ability changes when a different variety of texture measures is used. A second series of experiments used a larger variety of lengths of ESD and edge detector and slightly different elongations for the ESDs; the parameters of the feature detectors are shown in Table 4. A total of 39 operators were used; the texture images were the same as in Series I. The results of this experiment are summarized in Table 5. Of the 45 tex-

Table 1: Edge significances for D5-D84 (mica/raffia, Fig. 5) image.

	S	θ	E	L	ϕ	I	Overall
Top	0.674	0.664	0.935	0.735	0.542	2.418	5.968
Bottom	1.670	0.819	1.680	1.314	0.726	4.907	11.117
Left	0.176	0.465	0.377	0.216	0.341	0.211	1.785
Right	1.635	0.758	1.375	1.372	0.578	0.242	5.959

Table 2: Edge significances for D23-D94 (pebbles/bricks, Fig. 6) image.

	S	θ	E	L	ϕ	I	Overall
Top	0.025	0.551	0.063	0.272	0.136	2.682	3.730
Bottom	2.878	0.643	1.890	1.308	0.782	3.256	10.757
Left	1.104	0.582	0.310	1.068	0.380	1.880	5.324
Right	0.633	0.489	1.186	0.794	0.471	0.317	3.890

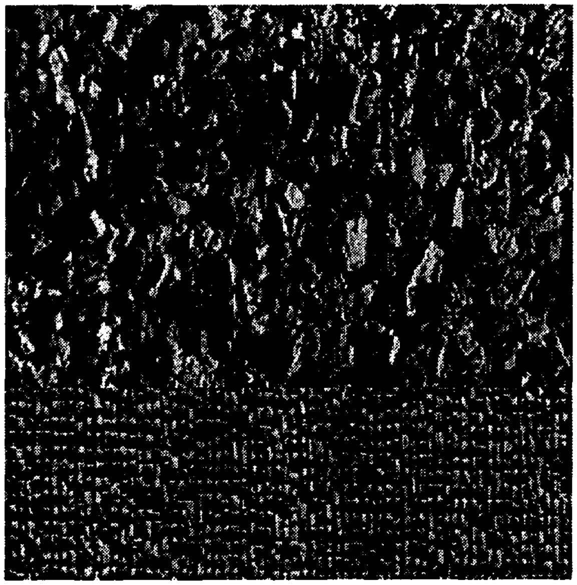


Figure 5: A texture edge constructed from Brodatz's images D5 (expanded mica) and D84 (raffia).

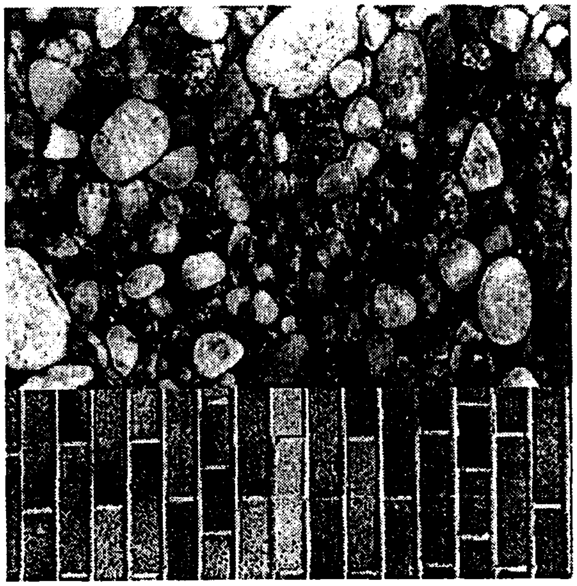


Figure 6: A texture edge formed from images D23 (pebbles) and D94 (bricks).

Table 3: Series I discrimination experiments

D9	D15	D23	D29	D35	D68	D84	D94	D98	
●●	●○	●○	●●	●○	●●	●●	●●	●●	D5 (Mica)
	○●	●●	●●	●●	●●	●●	●○	●●	D9 (Grass)
		●●	○●	●●	●○	●●	●●	●●	D15 (Straw)
			●●	○●	●●	●●	●●	●○	D23 (Pebbles)
				●●	●●	●●	●●	●●	D29 (Sand)
					●●	●●	●●	●●	D35 (Lizard skin)
						○●	●●	●○	D68 (Wood)
							●●	●●	D84 (Raffia)
● Corrcct discrimination								●●	D94 (Bricks)
○ Incorrect discrimination									D98 (Quartz)

Table 4: Feature detector parameters for discrimination experiments

	Elongated Structure Detectors			Edge Detectors		
	S	E	θ sep	L	ϕ sep	N
Series I	8, 12	0.3, 0.7, 1.0	45 deg	8,12	45 deg	4
Series II	4, 8, 12	0.3, 0.6, 1.0	45	6,9,12	45	3
Series III	8, 12	0.3, 0.7, 1.0	36	8,12	36	3

Table 5: Series II discrimination experiments

D9	D15	D23	D29	D35	D68	D84	D94	D98	
●●	●●	●●	●●	●●	●●	●●	●●	●○	D5 (Mica)
	○○	●●	○○	○●	●●	○●	●●	●●	D9 (Grass)
		●●	○●	○●	●○	●●	●●	●●	D15 (Straw)
			●●	●●	●●	●●	●●	●●	D23 (Pebbles)
				●●	●●	●●	●●	●●	D29 (Sand)
					●●	●●	●●	●●	D35 (Lizard skin)
						○●	●●	●●	D68 (Wood)
							●●	●●	D84 (Raffia)
● Correct discrimination								●●	D94 (Bricks)
○ Incorrect discrimination									D98 (Quartz)

Table 6: Series III discrimination experiments

D4	D20	D65	D67	D74	D87	D103	D110	D111	
●●	●●	●●	●●	●●	●●	●●	●●	●●	D1 (Wire weave)
	●●	●●	●●	●●	●●	●●	●●	●●	D4 (Cork)
		●●	●●	●●	●●	●●	●●	●●	D20 (Canvas)
			●●	●●	●●	●●	●●	●●	D65 (Rattan)
				●●	●●	●●	●●	●●	D67 (Plastic pellets)
					●●	●●	●●	●●	D74 (Coffee beans)
						●●	●●	●●	D87 (Sea fern)
							●○	●●	D103 (Burlap)
● Correct discrimination								●●	D110 (Paper)
○ Incorrect discrimination									D111 (Plastic bubbles)

ture pairs, 38 (84.4%) were correctly discriminated using all features including intensity, while 41 (91.1%) were correctly discriminated when intensity was ignored.

The third series of experiments used a larger variety of orientations of ESD and of edge detector, on a different set of 45 texture pairs, again from Brodatz; Table 4 contains the details. Table 6 summarizes the results of this experiment: 45 (100%) were correctly discriminated using all features including intensity, while 44 (97.7%) were correctly discriminated when intensity differences were ignored.

These empirical results indicate that these texture measures are able to reliably distinguish natural textures, for a variety of choices of the measures' parameters. However, one should not regard these measures as suitable for distinguishing all textures. The textures used in this series of experiments were all quite different. The current parameter values would probably not work well for discriminating textures that are more similar to one another, particularly fine-grain textures such as sand. For this purpose, a greater range of size and elongation resolution would be required. In addition, more than four or five orientations might be required to discriminate oriented textures differing by a small angle.

5.1 Comparison with other measures

The reader may object that the texture measures proposed here are complex and time-consuming to compute, and that simpler, more easily computable measures might perform just as well. For example, the texture measures proposed by Laws [20] are well known, efficiently computable, and have been shown to be effective in classifying textures [20,22]. For comparison purposes, Laws' texture features were tested on the same set of 45 texture pairs used in the first two series of discrimination experiments. Laws' method computes the "texture energy" of several texture features as follows. The image is convolved with each of several small kernels K_i, yielding feature images F_i. Each F_i is then convolved with a 15 × 15 pixel "texture energy" filter that computes an approximation to the variance in a window by summing the absolute values of the feature values. This operation produces several "texture energy" images T_i, which collectively form a vector-valued texture energy image. These vector values were used as input to the texture edge detector described above and in [27] to compute the significance of an edge.

The first series of tests used Laws' nine 3 × 3 masks. An edge operator with $N = 11$ slices was used to test for an edge. (The reader is referred to [20] for the details of these masks.) Table 5 summarizes the results: the texture edge was correctly located in 17 of the 45 texture images, or 37.3%; chance performance in this task is 25%.

This low level of performance may be due to the small size of the masks, which capture only fine-grain texture information; larger masks should be better able to capture larger-scale texture information, allowing better texture discrimination. Laws also uses a set of sixteen 5 × 5 kernels for texture classification. Four of these 5 × 5 masks (L5E5, L5S5, E5S5, and R5R5) have been claimed to be the most effective in classification tasks [22]. A second series of tests employed these 4 masks and their 90 degree rotations (a total of seven masks, since R5R5 is unchanged under rotation). The results, using an edge operator with $N = 9$ slices, are also shown in Table 5; 19 of the 45 images were correctly discriminated, or 42.2%. The larger masks allow for slightly better discrimination, but performance is still unsatisfactory.

These rather poor results may be attributed to a number of causes. First, the small kernels are sensitive to only fine-scale texture features, while the texture that distinguishes the image regions is characterized by larger-scale structures. If one must distinguish textures characterized by large-scale features, the efficiency of small masks must be sacrificed; there is no alternative but to measure large-scale features. Second, the 15 × 15 operator that measures the "texture energy" blurs the texture feature values across the edge. This effect causes the texture energy to change gradually across the edge, giving the intercepts of the fitted regression lines a high variance; this in turn causes the differences in intercepts to have lower significance. The feature detectors proposed in this paper are quite sensitive to position and are much less prone to the blurring effect, so they do not suffer from this loss of significance. An edge operator that assumed uniform texture fields on either side of the edge would average out the blurred feature values, leading to more effective discrimination[3], but such edge operators are not as useful in practice since texture images are rarely uniform.

6 Conclusion

In this paper, we have viewed texture as not merely pattern in the image, but as the projection of structure in the scene. We have broken the problem of texture segmentation into two phases, texture discrimination and localization of edges thus detected; this paper has focused on the former topic. To determine when two image textures differ, we must look beyond the image and infer parameters of the processes by which the textures were created, and try to determine whether those parameters differ. To this end, we must model texture in the scene, model its projection to the image, and determine a set of image features that allows us to discriminate reliably between physically different textures in the modeled class. The proposed texture model is general enough to model a large class of textured objects; it is based on distributions of shape, position and subtexture of texture substructures. Differences in simple image features derived from these distributions allow segmentation of physically different textures. The features used are size, elongation and orientation of elongated structures, length and orientation of image edges, and overall color differences between image regions. We use statistical tests to determine the significance of differences between estimated distributions of these image features. This method provides

[3] In fact, when such a test was carried out on the same set of texture pairs, using a multivariate t-test such as described in [27], both the nine 3 × 3 masks masks and the proposed texture measures discriminated the textures with 100% accuracy.

D9	D15	D23	D29	D35	D68	D84	D94	D98	
○○	○○	○○	○○	○○	●○	○○	●○	●○	D5 (Mica)
	●○	●●	○○	○○	○○	○○	○●	●●	D9 (Grass)
		○●	○○	○○	○○	○○	○●	●○	D15 (Straw)
			○●	○●	○●	○●	●●	●●	D23 (Pebbles)
				●○	●●	●●	○●	●●	D29 (Sand)
					●○	●○	●○	●○	D35 (Lizard skin)
						○●	○●	○○	D68 (Wood)
							○●	○●	D84 (Raffia)
								○○	D94 (Bricks)
									D98 (Quartz)

Table 7: Results using Laws' texture measures. First circle in each entry is for 3×3 operators, second is for 5×5 operators. Solid circle, correct discrimination; hollow, incorrect.

not only an indication of when textures differ, but also predicts how likely we are to be wrong if we infer the existence of a true discontinuity.

It must be emphasized again that no single set of image texture measures will suffice to discriminate between all physically different textures. Corresponding to any set of texture measures there are classes of textures that the measures cannot discriminate. The texture measures proposed here are capable of discriminating a large number of textures, but there are certainly many more that are indiscriminable. For example, human vision seems to discriminate some textures on the basis of terminations of elongated structures [14]; it might be desirable for a machine vision system to identify similar features. One must be very clear about the goals of a texture segmentation algorithm. If it is to discriminate just those textures that humans discriminate, then careful attention must be paid to psychophysical data that help to define the texture features employed by our visual system (e.g., [26,14,15]). If it is known that two classes of textures in the world must be discriminated, then image measures can be constructed for that purpose. In particular, if image structures are known to be well defined, then distributions of their attributes can be measured directly (e.g., [30]), and discrimination based on these directly-measured distributions will be superior to that based on more general measures such as those proposed here. Very fine grain textures, such as sand, may require texture measures such as intensity variance for discrimination. In the absence of special information about the domain, it is reasonable to measure image features similar to those measured by humans; this allows a machine vision system to detect those edges detected by humans, and prevents it from hallucinating edges not perceived by humans. The texture measures proposed here are similar to Julesz's 'textons' [14,15], and a texture segmentation program that uses these measures should have texture discrimination ability similar to that of humans.

Several problems await further research. The other aspect of the image segmentation problem, namely edge localization, is the subject of another report, and we shall not delve further into that subject here (but see [29]). The best values for the parameters of the feature detectors are currently under investigation. It might be useful to use some tuning method that makes slight changes to the parameters to obtain optimal performance over a set of example texture images. Further empirical tests must be performed in order to determine the ability of these texture measures to distinguish textures. However, the empirical results presented above are encouraging evidence for the effectiveness of these texture measures.

7 Acknowledgements

This research benefited from discussions with Tom Binford, Brian Wandell,and Andy Witkin. Ken Laws provided many helpful comments on an earlier draft. This research was supported by a subcontract to Advanced Decision Systems, S1093 S-1 (Phase II), "Knowledge Based Vision Task B," from a contract to the Defense Advanced Research Projects Agency.

References

[1] N. Ahuja and B.J. Schachter. Image models. *Comput. Surveys*, 13(4):373–397, 1981.

[2] J. Aloimonos and M. Swain. Shape from texture. In *Proc. Int. Joint Conf. on Art. Intell.*, pages 926–931, 1985.

[3] P. Brodatz. *Textures: A photographic album for artists and designers*. Dover, New York, 1966.

[4] T.N. Cornsweet. *Visual Perception*. Academic Press, New York, 1970.

[5] L.S. Davis and A. Mitiche. Edge detection in textures. *Comp. Graphics and Image Proc.*, 12(1):25–39, 1980.

[6] L.S. Davis and A. Mitiche. MITES: A model-driven, iterative texture segmentation algorithm. *Comp. Graphics and Image Proc.*, 19:95–110, 1982.

[7] A. Gagalowicz and S.D. Ma. Model driven synthesis of natural textures for 3-D scenes. *Comput. and Graphics*, 10(2):161–170, 1986.

[8] A. Gagalowicz and S.D. Ma. Sequential synthesis of natural textures. *Comp. Vision, Graphics, and Image Proc.*, 30:289–315, 1985.

[9] R.M. Haralick. Digital step edges from zero crossing of second directional derivatives. *IEEE Trans. Pattern Anal. Machine Intell.*, PAMI-6(1):58–68, 1984.

[10] R.M. Haralick. Edge and region analysis for digital image data. *Comp. Graphics and Image Proc.*, 12:60–73, 1980.

[11] R.M. Haralick. Statistical and structural approaches to texture. *Proc. IEEE*, 67(5):786–804, 1979.

[12] G. Healey and T.O. Binford. A color metric for computer vision. In *Proc. IEEE Conf. on Comp. Vision and Patt. Recog.*, 1988. Submitted.

[13] G. Healey and T.O. Binford. The role and use of color in a general vision system. In *Image Understanding Workshop*, pages 599–613, 1987.

[14] B. Julesz. Textons, the elements of textural perception, and their interactions. *Nature*, 290:91–97, 1981.

[15] B. Julesz and J.R. Bergen. Textons, the fundamental elements in preattentive vision and perception of textures. *Bell System Tech. J.*, 62(6):1619–1645, 1983.

[16] K.-I. Kanatani and T.-C. Chou. *Shape from texture: General principle.* Technical Report CAR-TR-184, Center for Automation Research, Univ. of Maryland, February 1986.

[17] R.L. Kashyap. Image models. In T.Y. Young and K.-S. Fu, editors, *Handbook of Pattern Recognition and Image Processing, pages 281–310, Academic Press*, Orlando, Fla., 1986.

[18] B. Kjell and C.R. Dyer. *Edge separation and orientation texture measures.* Technical Report 559, Computer Science Dept, Univ. of Wisconsin, Madison, October 1984.

[19] K.I. Laws. *Goal-directed textured-image segmentation.* Technical Report 334, AI Center, SRI International, September 1984.

[20] K.I. Laws. *Textured image segmentation.* Technical Report UCSIPI 940, USC Image Proc. Institute, Univ. of Southern Calif., 1980.

[21] Y. Leclerc. Capturing the local structure of image discontinuities in two dimensions. In *Proc. IEEE Conf. on Comp. Vision and Patt. Recog.*, pages 34–38, 1985.

[22] M. Pietikäinen, A. Rosenfeld, and L.S. Davis. Experiments with texture classification using averages of local pattern matches. *IEEE Trans. Syst., Man and Cybern.*, SMC-13(3):421–426, 1983.

[23] H.M. Raafat and A.K.C. Wong. Texture-based image segmentation. In *Proc. IEEE Conf. on Comp. Vision and Patt. Recog.*, pages 469–475, 1986.

[24] L. Sachs. *Applied Statistics.* Springer-Verlag, New York, 1984.

[25] A. Treisman. A feature-integration theory of attention. *Cognitive Psychology*, 12:97–136, 1980.

[26] A. Treisman. Preattentive processing in vision. *Comp. Vision, Graphics, and Image Proc.*, 31(2):156–177, 1985.

[27] R. Vistnes. Computer texture analysis and segmentation. Submitted, 1987. An abridged version appears in *Proceedings, IEEE Conf. Comp. Vision*, Miami, 1987.

[28] R. Vistnes. *Computer texture analysis and segmentation.* PhD thesis, Computer Science Dept., Stanford University, June 1988.

[29] R. Vistnes. Texture edge localization. In *Image Understanding Workshop*, 1988 (these proceedings).

[30] H. Voorhees and T. Poggio. Detecting textons and texture boundaries in natural images. In *Proc. Int. Conf. on Comp. Vision*, pages 250–258, 1987.

[31] A.P. Witkin. Recovering surface shape and orientation from texture. *Artificial Intelligence*, 17:17–45, 1981.

Model Driven Edge Detection*

Pascal Fua and Yvan G. Leclerc

Artificial Intelligence Center, SRI International
333 Ravenswood Avenue, Menlo Park, California 94025

Abstract

Standard edge-detectors fail to find most relevant edges, finding either too many or too few, because they lack a geometrical model to guide their search. We present a technique that integrates both photometric and geometric models with an initial estimate of the boundary. The strength of this approach lies in the geometric model's ability to overcome various photometric anomalies, thereby finding boundaries that could not otherwise be found. Furthermore, the model can be used to score edges based on their goodness of fit, thus allowing one to use the semantic model information to accept or reject them.

1 Introduction

In real-world images, object boundaries cannot be detected solely on the basis of their photometry because of the presence of noise and various photometric anomalies. Thus, all methods for finding boundaries based on purely local statistical criteria are bound to make mistakes, finding either too many or too few edges, usually based on arbitrary thresholds.

To supplement the weak and noisy local information, we consider the geometrical constraints that object models can provide. We do this by describing a boundary as an elastic curve with a deformation energy derived from the geometrical constraints, as suggested by Terzopoulos and Kass et al. [4,7]. Local minima in this energy correspond to boundaries that exactly match the object model. We incorporate the photometric constraints by defining a photometric energy as the average along the curve of a scalar energy field derived from the photometric model of an edge. Local minima in this energy correspond to boundaries that exactly match the photometric model. We find a candidate boundary by deforming the curve in such a way as to minimize its total energy, which is the sum of the deformation and photometric energies. We call this deformation process "optimizing" the curve. Once a curve has been optimized, i.e., once it has settled in a local minimum of the total energy (which is, in effect, a compromise between the two constraints), we can then use the object models more fully to determine whether the curve actually corresponds to an object boundary.

Such "energy-minimizing curves" have two key advantages:

- The geometric constraints are directly used to guide the search for a boundary.
- The edge information is integrated along the entire length of the curve, providing a large support *without including the irrelevant information off of the curve.*

Taken together, these advantages allow energy-minimizing curves to find photometrically weak boundaries that local edge detectors simply could not find without also finding many irrelevant boundaries.

As originally presented by Terzopoulos and Kass [4,7], energy-minimizing curves were basically designed for an interactive environment. In this paper, we extend these ideas so that energy-minimizing curves can be more easily embedded in fully automated vision systems, and so that we can be confident that optimized curves are good candidates for object boundaries.

In the next section, we describe a photometric edge model based on a standard definition of a step-edge and show that the local minima of an energy-minimizing curve approximate this criterion accurately in more general conditions. Next, we derive the deformation energy and discuss our implementation for the simple case of smooth curves. We then describe an application of these smooth curves to delineate roads and the use of more global geometric constraints in the case of rectilinear objects such as buildings in aerial images.

2 A Photometric Model: Step-Edges

Haralick [3], Canny [1], and Torre and Poggio [6] define step-edge points as those for which the first directional derivative of image intensity, in the direction of the gradient, is extremal. That is, an edge point (x_0, y_0) satisfies

$$\left.\frac{\partial^2 \mathcal{I}(x,y)}{\partial \mathbf{g}_0^2}\right|_{(x,y)=(x_0,y_0)} = 0, \qquad (1)$$

where

$$\begin{aligned}\mathbf{g}_0 &= \mathbf{g}(x_0,y_0) = \frac{\nabla \mathcal{I}(x_0,y_0)}{|\nabla \mathcal{I}(x_0,y_0)|},\\ \frac{\partial}{\partial \mathbf{g}_0} &= \mathbf{g}_0 \cdot \nabla,\end{aligned}$$

and $\mathcal{I}(x,y)$ represents the image intensities after convolution with a twice-differentiable kernel (typically a Gaussian).[1] It can be shown that Eq. (1) is equivalent to

$$\left.\frac{\partial\,|\nabla \mathcal{I}(x,y)|}{\partial \mathbf{g}_0}\right|_{(x,y)=(x_0,y_0)} = 0.$$

In other words, the Haralick et al. criterion for an ideal step-edge point is equivalent to the criterion that the magnitude of the image intensity gradient be maximal in the gradient direction.

*The work reported here was partially supported by the Defense Advanced Research Projects Agency under contracts DACA76-85-C-0004 and MDA903-86-C-0084. We wish to thank Demetri Terzopoulos for the many discussions we have had concerning this work.

[1] This definition is a good approximation to the position of discontinuities in the underlying intensity function when the diameter of support of the operator is much smaller than the radius of curvature of the edge. In particular, it is not applicable at corners and junctions of edges.

However, for noisy non ideal step-edges the gradient direction may become very unreliable, especially at some distance from the edge itself. We therefore use the following, somewhat more general, definition of an edge:

Definition 1 *An edge is a curve $\mathcal{C}$ whose points have a gradient magnitude that is maximal in the direction normal to the curve. That is, all points along the curve (called edge-points) satisfy*

$$\frac{\partial\,|\nabla\mathcal{I}(\mathbf{f}(s))|}{\partial\mathbf{n}(\mathbf{f}(s))} = 0, \tag{2}$$

where s is the arc-length of $\mathcal{C}$, $\mathbf{f}(s)$ is a vector function mapping the arc-length s to points (x,y) in the image, and $\mathbf{n}(\mathbf{f}(s))$ is the normal to $\mathcal{C}$.

Note that this definition is equivalent to the Haralick et al. definition for ideal noise-free step-edges.

To incorporate this photometric model in the energy-minimizing curve approach, we define the photometric energy of the curve $\mathcal{C}$ as

$$\mathcal{E}_P(\mathcal{C}) = \frac{-\int|\nabla\mathcal{I}(\mathbf{f}(s))|\,ds}{|\mathcal{C}|},$$

where $|\mathcal{C}|$ is the length of $\mathcal{C}$.

To understand why we have chosen this particular energy function, consider the following. We prove in the Appendix that, when $\mathcal{C}$ is either an open or closed curve that has been optimized, i.e., when it is a local minimum of $\mathcal{E}_P$ with respect to infinitesimal deformations of the curve,

$$\frac{\partial|\nabla\mathcal{I}(\mathbf{f}(s))|}{\partial\mathbf{n}(\mathbf{f}(s))} = \gamma(s)\left(|\nabla\mathcal{I}(\mathbf{f}(s))| - \frac{\int|\nabla\mathcal{I}(\mathbf{f}(s))|\,ds}{|\mathcal{C}|}\right), \tag{3}$$

where $\gamma(s)$ is the curvature of $\mathcal{C}$. Furthermore, we prove that when $\mathcal{C}$ is an open curve that has been optimized,

$$|\nabla\mathcal{I}(\mathbf{f}(0))| = |\nabla\mathcal{I}(\mathbf{f}(|\mathcal{C}|)| = \frac{\int|\nabla\mathcal{I}(\mathbf{f}(s))|\,ds}{|\mathcal{C}|} \tag{4}$$

also holds.

These two equations have the following consequences. Eq. (3) implies that the points along an optimized curve all are edge-points when either the curvature is zero (i.e., it is a straight-line segment), or the magnitude of the gradient along the curve is constant. Consequently, optimized curves whose curvature is small or for which the magnitude of the gradient is approximately constant are good approximations to edges. In particular, we have found that replacing $|\nabla\mathcal{I}(x,y)|$ by $\log|\nabla\mathcal{I}(x,y)|$ (which has no effect for ideal step-edges) yields more stable solutions because the second term in Eq. (3) is smaller for non constant edge strengths, and hence the approximation is generally better.

For an open-ended curve, Eq. (4) must also be satisfied. This equation implies that the curve is stable only when the gradient magnitude at the end-points equals the average. In other words, an open-ended energy-minimizing curve placed exactly along an edge will shrink or expand (following the edge!) until this condition is met. When the gradient magnitude is constant, of course, this condition is always met, and curves therefore remain unchanged. Thus, again, using $\log|\nabla\mathcal{I}(x,y)|$ yields more stable solutions because this equation is more nearly satisfied for non constant edge strengths. An important point is that if we had chosen to simply use the integral of gradient magnitude, as opposed to the average, this equation would not hold and curves would simply expand without bound!

After optimization, candidate edges can be scored on how well they match the photometric model by computing the proportion of points along the curve that satisfy Eq. (2) to within some tolerance. We use this score in our applications to accept or reject any given edge.

3 A Geometric Model: Smooth Curves

3.1 Theory

We have seen in the previous section that energy-minimizing curves match edges well wherever they have a low curvature. To ensure stable results, we define the deformation energy of such curves so that their curvature remains small and the minimum energy state is close to the photometric edge. This can be achieved simply by defining the deformation energy of a curve $\mathcal{C}$ as

$$\mathcal{E}_D(\mathcal{C}) = \int\gamma(s)^2\,ds,$$

where $\gamma(s)$ is the curvature of $\mathcal{C}$. The total energy of such a curve is then $\mathcal{E}(\mathcal{C}) = \mathcal{E}_P(\mathcal{C}) + \lambda\mathcal{E}_D(\mathcal{C})$, where $\mathcal{E}_P(\mathcal{C})$ is the photometric energy defined in the previous section and $\mathcal{E}_D(\mathcal{C})$ the deformation energy described above.

To delineate a photometric edge using one of these curves, one must provide an initial estimate of the location of the curve and then optimize it using the procedure described in the next subsection. The choice of λ is directly related to the roughness of the initial estimate as follows. Let $\mathcal{C}_i$ be the initial curve and $\mathcal{C}_f$ be the curve after optimization. Let δ_i denote the operator

$$\delta_i = \left.\frac{\delta}{\delta\mathcal{C}}\right|_{\mathcal{C}=\mathcal{C}_i};$$

that is, δ_i is the variational operator evaluated at the initial curve. Define δ_f similarly for the final curve. Since, by definition, $\delta_f\mathcal{E}(\mathcal{C}) = 0$, then

$$\lambda = \frac{|\delta_f\,\mathcal{E}_P(\mathcal{C})|}{|\delta_f\,\mathcal{E}_D(\mathcal{C})|}.$$

When the initial estimate is known to be close to the final answer, we choose

$$\lambda = \frac{|\delta_i\,\mathcal{E}_P(\mathcal{C})|}{|\delta_i\,\mathcal{E}_D(\mathcal{C})|}$$

so that we remain as close to the initial estimate as possible. If instead we want to smooth the initial estimate, we can use higher values of λ. This is in fact the simplest way of imposing a geometric model, in this case smoothness, upon the data.

3.2 Implementation

In the actual implementation, the curves are described as polygons with n equidistant vertices $X = \{(x_i\ y_i),\ i = 1,\ldots,n\}$ and the deformation energy can be discretised as

$$\mathcal{E}_D = \sum_i(2x_i - x_{i-1} - x_{i+1})^2 + (2y_i - y_{i-1} - y_{i+1})^2 \tag{5}$$

when $\gamma(s)$ is small. To perform the optimization we could use a simple gradient descent technique as we have already suggested

[5], but it would be extremely slow for curves with a large number of vertices. Instead, we embed the curve in a viscous medium and solve the equation of dynamics

$$\frac{\partial \mathcal{E}}{\partial X} + \alpha \frac{dX}{dt} = 0,$$

where $\mathcal{E} = \mathcal{E}_P + \lambda \mathcal{E}_D$ and α is the viscosity of the medium [7]. Because the deformation energy $\mathcal{E}_D$ in Eq. 5 is quadratic, its derivative with respect to X is linear, and therefore

$$\frac{\partial \mathcal{E}_D}{\partial X} = KX,$$

where K is a pentadiagonal matrix. Thus, each iteration of the optimization amounts to solving the linear equation

$$KX_t + \alpha(X_t - X_{t-1}) = \left.\frac{\partial \mathcal{E}_P}{\partial X}\right|_{X_{t-1}} .$$

We start with an initial step size δ and compute the viscosity so that

$$\alpha = \frac{\sqrt{2n}}{\delta} \left|\frac{\partial \mathcal{E}}{\partial X}\right|,$$

where n is the number of vertices. This ensures that the initial displacement of each vertex is on the average of magnitude δ. Because of the non linear term, we must verify that the energy has decreased from one iteration to the next. If, instead, the energy has increased, the curve is reset to its previous position, the step size is decreased, and the viscosity recomputed accordingly. This is repeated until the step size becomes less than some threshold value. In most cases, because of the presence of the linear term that propagates constraints along the whole curve in one iteration, only a very small number of iterations are required to optimize the initial curve. For example, going from the initial estimate of the closed curve with about 50 vertices shown in Figure 1(a) to the optimized result shown in Figure 1(b) took fewer than 20 iterations. Even fewer iterations were required for the open curve.

4 Applications

In this section we present two applications of our energy-minimizing curves in an interactive context where the user provides a rough initial estimate of the location of the contours. The energy-minimizing curve approach has also been incorporated in a fully automated system that extracts buildings and roads from aerial imagery by using the boundaries of a region segmentation to generate the initial estimates. This application is described in detail in another paper also appearing in these proceedings [2].

4.1 Road Delineation

We can directly apply the smooth curve model of the previous section to the problem of finding road boundaries in aerial images. The boundaries of such roads can be modeled as smooth curves that are approximately parallel. We therefore represent a road instance by a smooth polygonal curve forming the center of the road, and we associate with each vertex i of this skeleton a width w_i that defines the two curves that are the candidate road boundaries. We define the photometric energy as the sum of the photometric energies of the two boundary curves, and the deformation energy as the sum of the smoothness term described above and an additional term that enforces the parallel constraint:

$$\sum_i (w_i - w_{i-1})^2.$$

We have applied this model to two road-segments in the aerial image of Figure 2(a). Figure 2(b) shows the initial estimates for two road-segment boundaries and Figure 2(c) shows the final optimized boundaries.

This image is an excellent example of a situation where it is difficult to use a standard local edge operator, such as the Canny operator, to find the relevant boundaries. Because the roads are dirt roads with ill-defined edges, no single edge-strength threshold yields a satisfactory set of boundaries: when the threshold is too high, most of the road edges are lost, whereas when the threshold is too low, there is a plethora of irrelevant edges (see Figure 3). Furthermore, the road boundaries are completely non existent at the intersections. This is the kind of situation in which the predictive power of the geometric model, as captured by the energy-minimizing curve, is crucial.

4.2 Building Delineation

Local geometric constraints, such as smoothness, are insufficient when we wish to find the boundaries of more complex objects. In such case, we need more global constraints. For example, to find the boundaries of rectilinear objects (whose boundary is composed of straight line-segments forming a closed region intersecting at right angles), we need to add a deformation energy term that enforces rectilinearity. To delineate such objects, we use a set of straight edges which are polygonal curves with only two vertices and the additional energy term

$$\mathcal{E}_D = \sum_i ((\theta_i - \theta) \bmod \pi/2)^2 ,$$

where θ_i is the angle of an edge and θ the average direction of all edges modulo $\pi/2$.

Figure 4(a) shows three initial sets of edges that have been drawn by hand but could have been discovered by an edge detector. Two of them correspond to actual buildings and one to a tree. Figure 4(b) shows the final optimized results after imposing the rectilinear constraint. In Figure 4(c), for each edge, only the pixels that are edge-points according to Definition 1 are shown. While most of the pixels belonging to building edges satisfy the edge criterion, very few of those belonging to tree boundaries do. Computing the proportion of edge-points in the optimized edges thus gives us an efficient way to measure how well the photometric boundaries match the model geometry. This criterion is used extensively in the automated system described in [2] to decide whether or not boundaries of a given shape match the actual image photometry.

5 Summary and Conclusion

We have presented a technique for finding object boundaries that integrates both photometric and geometric models with an initial estimate of the boundary. The models are incorporated by defining an energy function for curves that is minimal when the models are exactly satisfied. The initial estimate is used as the starting point for finding a local minimum of this energy function by embedding the initial curve in a viscous medium and solving the equations of dynamics.

The strengths of this "energy-minimizing curve" approach are that the geometric constraints are directly used to guide the search for a boundary and that the edge information is integrated along the entire length of the curve, thereby providing a large support.

We have shown the precise relationship between optimized curves and a standard definition of an edge, and discussed how this relationship can be used to determine when an optimized curve should be accepted as a candidate edge for further processing. We have also presented methods for automatically choosing the optimization parameters and how they should change over time. We found that these contributions were crucial to our ability to embed energy-minimizing curves in the automated system presented in [2] and that they have provided us with the ability to find photometrically weak boundaries that local edge detectors simply could not find without also finding many irrelevant boundaries.

Appendix

In this appendix, we show that a necessary and sufficient condition for an open curve C to be a local minimum of

$$\mathcal{E}(C) = \frac{\int G(\mathbf{f}(s))ds}{\int ds} \tag{6}$$

with respect to all infinitesimal deformations is that

$$\frac{dG(\mathbf{f}(s))}{d\mathbf{n}(s)} = \gamma(s)\left[G(\mathbf{f}(s)) - \frac{\int G(\mathbf{f}(s))ds}{\int ds}\right] \tag{7}$$

and

$$G(\mathbf{f}(0)) = G(\mathbf{f}(|C|)) = \frac{\int G(\mathbf{f}(s))ds}{\int ds}. \tag{8}$$

where the curve C is parameterized by its arc-length s; $\mathbf{f}(s)$ is a *vector function mapping the arc-length* s *to points* (x, y) along the curve; $\mathbf{n}(s)$ is the normal to the curve; $\mathbf{t}(s)$ is the tangent to the curve; and $\gamma(s)$ is its curvature. Furthermore, Eq. (7) alone is *the necessary and sufficient condition for closed curves.*

To prove this result, consider deformations of the curve C, which we shall call C_λ, such that the mapping from arc-length s to points (x, y) is of the form

$$\mathbf{f}_\lambda(s) = \mathbf{f}(s) + \lambda\left(\eta(s)\mathbf{n}(s) + \tau(s)\mathbf{t}(s)\right). \tag{9}$$

where $\eta(s)$ and $\tau(s)$ are arbitrary continuous and differentiable function. When C is a closed curve, $\eta(s)$ and $\tau(s)$ are constrained to be periodic of period $|C|$. Since C is a local extremum of $\mathcal{E}(C)$ if and only if

$$\left.\frac{d\mathcal{E}(C_\lambda)}{d\lambda}\right|_{\lambda=0} = 0, \tag{10}$$

for all $\eta(s)$ and $\tau(s)$, we now prove the following theorem.

Theorem: Eq. (10) holds for all $\eta(s)$ and $\tau(s)$ if and only if eq. (7) and (8) are satisfied.

Proof: By definition,

$$\mathcal{E}(C_\lambda) = \frac{\int G(\mathbf{f}_\lambda(s_\lambda))ds_\lambda}{\int ds_\lambda}. \tag{11}$$

where s_λ is the arc-length of C_λ. We can rewrite this in terms of s by using the equality

$$ds_\lambda = |\mathbf{f}'_\lambda(s)|\,ds. \tag{12}$$

Thus,

$$\mathcal{E}(C_\lambda) = \frac{\int G(\mathbf{f}_\lambda(s))\,|\mathbf{f}'_\lambda(s)|\,ds}{\int |\mathbf{f}'_\lambda(s)|\,ds}. \tag{13}$$

and therefore, to within the non zero factor of $1/(\int |\mathbf{f}'_\lambda(s)|\,ds)^2$:

$$\begin{aligned}
\frac{d\mathcal{E}(C_\lambda)}{d\lambda} &= \frac{d}{d\lambda}\left[\int G(\mathbf{f}_\lambda(s))\,|\mathbf{f}'_\lambda(s)|\,ds\right]\int |\mathbf{f}'_\lambda(s)|\,ds \\
&\quad - \left[\int G(\mathbf{f}_\lambda(s))\,|\mathbf{f}'_\lambda(s)|\,ds\right]\frac{d}{d\lambda}\int |\mathbf{f}'_\lambda(s)|\,ds \\
&= \int \frac{dG(\mathbf{f}_\lambda(s))}{d\lambda}\,|\mathbf{f}'_\lambda(s)| + G(\mathbf{f}_\lambda(s))\frac{d\,|\mathbf{f}'_\lambda(s)|}{d\lambda}ds \int |\mathbf{f}'_\lambda(s)|\,ds \\
&\quad - \left[\int G(\mathbf{f}_\lambda(s))\,|\mathbf{f}'_\lambda(s)|\,ds\right]\int \frac{d\,|\mathbf{f}'_\lambda(s)|}{d\lambda}ds.
\end{aligned} \tag{14}$$

The first term we need to evaluate in Eq. (9) is $d\,|\mathbf{f}'_\lambda(s)|\,/d\lambda$, for which we need the following equalities:

$$\begin{aligned}
\frac{d\mathbf{f}(s)}{ds} &= \mathbf{f}'(s) = \mathbf{t}(s) \\
\frac{d\mathbf{t}(s)}{ds} &= \mathbf{t}'(s) = \gamma(s)\mathbf{n}(s) \\
\frac{d\mathbf{n}(s)}{ds} &= \mathbf{n}'(s) = -\gamma(s)\mathbf{t}(s).
\end{aligned}$$

Therefore,

$$\begin{aligned}
\mathbf{f}'_\lambda(s) &= \mathbf{f}'(s) + \lambda\left[\eta'(s)\mathbf{n}(s) + \eta(s)\mathbf{n}'(s) + \tau'(s)\mathbf{t}(s) + \tau(s)\mathbf{t}'(s)\right] \\
&= \mathbf{t}(s) + \lambda\left[\eta'(s)\mathbf{n}(s) - \eta(s)\gamma(s)\mathbf{t}(s)\right. \\
&\quad \left.+\tau'(s)\mathbf{t}(s) + \tau(s)\gamma(s)\mathbf{n}(s)\right] \\
&= \mathbf{t}(s)\left[1 + \lambda(\tau'(s) - \eta(s)\gamma(s)\right] \\
&\quad +\mathbf{n}(s)\left[\lambda(\eta'(s) + \tau(s)\gamma(s))\right].
\end{aligned} \tag{15}$$

Since $\mathbf{t}(s)$ and $\mathbf{n}(s)$ are, by definition, orthogonal unit vectors, we have

$$|\mathbf{f}'_\lambda(s)| = \sqrt{[1 + \lambda(\tau'(s) - \eta(s)\gamma(s))]^2 + [\lambda(\eta'(s) + \tau(s)\gamma(s))]^2}. \tag{16}$$

and therefore $d\,|\mathbf{f}'_\lambda(s)|\,/d\lambda =$

$$\frac{[1 + \lambda(\tau'(s) - \eta(s)\gamma(s))](\tau'(s) - \eta(s)\gamma(s)) + \lambda(\eta'(s) + \tau(s)\gamma(s))^2}{\sqrt{[1 + \lambda(\tau'(s) - \eta(s)\gamma(s))]^2 + [\lambda(\eta'(s) + \tau(s)\gamma(s))]^2}}. \tag{17}$$

The second term we need to evaluate is

$$\begin{aligned}
\frac{dG(\mathbf{f}_\lambda(s))}{d\lambda} &= (\eta(s)\mathbf{n}(s) + \tau(s)\mathbf{t}(s)) \cdot \nabla G(\mathbf{f}_\lambda(s)) \\
&= \eta(s)\frac{dG(\mathbf{f}_\lambda(s))}{d\mathbf{n}(s)} + \tau(s)\frac{dG(\mathbf{f}_\lambda(s))}{d\mathbf{t}(s)}.
\end{aligned} \tag{18}$$

Substituting Eq. (16), (17), and (18) into Eq. (11), evaluating at $\lambda = 0$, and multiplying by the non-zero factor $1/(\int |\mathbf{f}'_\lambda(s)|\,ds)^2$,

$$\begin{aligned}
\left.\frac{d\mathcal{E}(C_\lambda)}{d\lambda}\right|_{\lambda=0} &= 0 \\
&= \int ds \int \eta(s)\frac{dG(\mathbf{f}(s))}{d\mathbf{n}(s)} + \tau(s)\frac{dG(\mathbf{f}(s))}{d\mathbf{t}(s)} \\
&\quad + G(\mathbf{f}(s))(\tau'(s) - \eta(s)\gamma(s))ds \\
&\quad - \int G(\mathbf{f}(s))ds \int (\tau'(s) - \eta(s)\gamma(s))ds.
\end{aligned}$$

Dividing by $\int ds$ and rearranging terms, we have

$$\begin{aligned} 0 \;=\; & \int \eta(s)\left[\frac{dG(\mathbf{f}(s))}{d\mathbf{n}(s)} - \gamma(s)\left(G(\mathbf{f}(s)) - \frac{\int G(\mathbf{f}(s))ds}{\int ds}\right)\right] ds \\ & + \int \tau(s)\frac{dG(\mathbf{f}(s))}{d\mathbf{t}(s)} + \tau'(s)\left[G(\mathbf{f}(s)) - \frac{\int G(\mathbf{f}(s))ds}{\int ds}\right] ds. \end{aligned}$$

A necessary and sufficient condition for the first integral to be zero for all $\eta(s)$ is that the term multiplying $\eta(s)$ be zero, which is Eq. (7).

Integrating the second integral by parts:

$$\begin{aligned} 0 \;=\; & \int \tau(s)\frac{dG(\mathbf{f}(s))}{d\mathbf{t}(s)}ds + \tau(s)\left(G(\mathbf{f}(s)) - \frac{\int G(\mathbf{f}(s))ds}{\int ds}\right)\Bigg|_0^{|\mathcal{C}|} \\ & - \int \tau(s)\frac{dG(\mathbf{f}(s))}{d\mathbf{t}(s)}ds, \end{aligned} \tag{19}$$

therefore,

$$\begin{aligned} 0 \;=\; & \tau(|\mathcal{C}|)\left(G(\mathbf{f}(|\mathcal{C}|)) - \frac{\int G(\mathbf{f}(s))ds}{\int ds}\right) \\ & -\tau(0)\left(G(\mathbf{f}(0)) - \frac{\int G(\mathbf{f}(s))ds}{\int ds}\right). \end{aligned} \tag{20}$$

A necessary and sufficient condition for this equality to hold for all $\tau(s)$ is that both terms be zero, which is Eq. (8).

Thus, a necessary and sufficient condition for an open curve $\mathcal{C}$ to be a local extremum of $\mathcal{E}(\mathcal{C})$ is that both Eq.(7) and (8) hold.

Furthermore, for closed curves, $\tau(0) \equiv \tau(|\mathcal{C}|)$ and $G(\mathbf{f}(0)) \equiv G(\mathbf{f}(|\mathcal{C}|))$; hence Eq. (20) is always satisfied. Therefore, a necessary and sufficient condition for a closed curve $\mathcal{C}$ to be a local extremum of $\mathcal{E}(\mathcal{C})$ is that Eq. (7) holds.

Q.E.D.

References

[1] J. Canny, "A Computational Approach to Edge Detection," *IEEE Trans. PAMI*, **8**(6), pp. 679-698, 1986.

[2] P. Fua and A. Hanson, "Automated Extraction of Generic Shapes Using Model-Based Optimization," *International Journal of Computer Vision*, submitted to this issue, 1988.

[3] R.M. Haralick, "Digital Step Edges from Zero Crossings of Second Directional Derivatives," *IEEE Trans. PAMI*, **6**(1), pp. 58-68, 1984.

[4] M. Kass, A. Witkin and D. Terzopoulos, "Active Contour Models," *International Journal of Computer Vision*, 4, 1987.

[5] Y.G. Leclerc and P. Fua, "Finding Object Boundaries Using Guided Gradient Ascent," *Topical Meeting on Machine Vision, Technical Digest Series, Optical Society of America*, Washington, D.C., **12**, pp. 168-161, 1987. Also presented at the DARPA Image Understanding Workshop, Los Angeles, California, February, 1987.

[6] V. Torre and T. Poggio, "On Edge Detection," *IEEE Trans. PAMI*, **8**(2), pp. 147-163, 1986.

[7] D. Terzopoulos, "On Matching Deformable Models to Images," *Topical Meeting on Machine Vision, Technical Digest Series, Optical Society of America*, Washington, D.C., **12**, pp. 160-167, 1987.

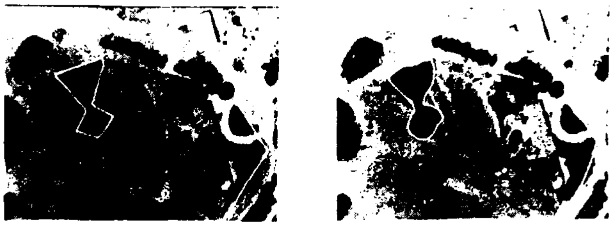

(a) (b)

Figure 1: (a) Initial estimates for both an open and a closed boundary.
(b) Final outlines after optimization.

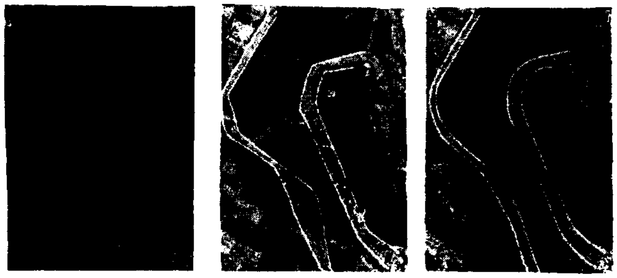

(a) (b) (c)

Figure 2: (a) An aerial image showing a network of dirt roads.
(b) Two initial estimates of the road outlines.
(c) Final delineations after optimization.

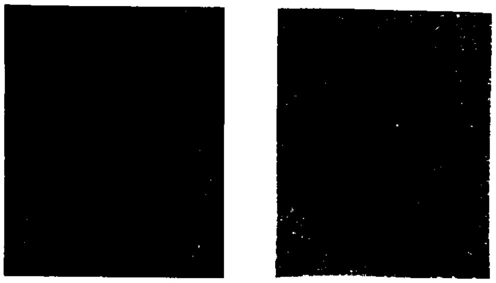

(a) (b)

Figure 3: Edge images generated by the Canny edge detector run on the dirt road image with two different sets of edge strength thresholds. Note that when the thresholds are too high, as in (a), most edges are lost, while many irrelevant edges are found when the thresholds are dropped, as in (b).

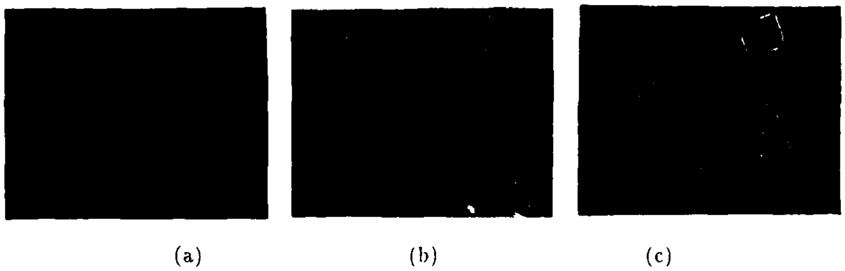


Figure 4: (a) Three sets of edges overlaid on an aerial image. (b) The edges after optimization using a rectilinearity constraint. (c) The pixels in all edges that satisfy our definition of an edge pixel. Note that while most pixels in the edges that are building edges satisfy our criterion, almost none of those that belong to tree edges do.

GENERALIZING EPIPOLAR-PLANE IMAGE ANALYSIS ON THE SPATIOTEMPORAL SURFACE*

H. Harlyn Baker
Robert C. Bolles

Artificial Intelligence Center
SRI International
333 Ravenswood Avenue
Menlo Park, CA 94025.

Abstract

The previous implementations of our Epipolar-Plane Image Analysis mapping technique demonstrated the feasibility and benefits of the approach, but were carried out for restricted camera geometries. The question of more general geometries made utility for autonomous navigation uncertain. We have developed a generalization of the analysis that a) enables varying view direction, including varying over time, b) provides three-dimensional connectivity information for building coherent spatial descriptions of observed objects, and c) operates sequentially, allowing initiation and refinement of scene feature estimates while the sensor is in motion. To implement this generalization it was necessary to develop an explicit description of the evolution of images over time. We have achieved this by building a process that creates a set of two-dimensional manifolds defined at the zeroes of a three-dimensional spatiotemporal Laplacian. These manifolds represent explicitly both the spatial and temporal structure of the temporally-evolving imagery, and we term them *spatiotemporal surfaces*. The surfaces are constructed incrementally, as the images are acquired. We describe a tracking mechanism that operates locally on these evolving surfaces in carrying out three-dimensional scene reconstruction.

1. INTRODUCTION

1.1. Epipolar-Plane Image Analysis

In [2], we described a sequence analysis technique we developed for use in obtaining depth estimates for points in a static scene. The approach bridged the usual dichotomy of depth sensing in that its large number of images led to a large baseline and thus high accuracy, while rapid image sampling gave minimal change from frame to frame, eliminating the correspondence problem. Rather than choosing quite disparate views and putting features into correspondence by stereo matching, with this technique we chose to process massive amounts of similar data, but with much *simpler and more robust techniques*. The technique capitalized on several constraints we could impose on the image acquisition process, namely:

1. the camera moved along a linear path:
2. it acquired images at equal spacing as it moved:
3. the camera's view was orthogonal to its direction of travel.

With these constraints, we could guarantee that

1. individual scene features would be observed in single epipolar planes over the period of scanning;
2. images of these planes could be constructed by collecting corresponding image scanlines in successive frames;
3. the motion of scene features in these images would appear as linear tracks.

We termed these image planes *epipolar-plane images*, or EPIs, and the process *Epipolar-Plane Image Analysis*.

1.2. Problems with the Previous Approach

In the earlier publication we commented on our principal dissatisfactions with the approach, and the limitations which would restrict its usefulness. Summarizing, these were:

Limitations:

L_1 Orthogonal viewing would preclude many of the camera attitudes one would expect to be necessary for an autonomous vehicle — noticeably that attitude in which the vehicle is looking along its direction of motion, or when it is to track some particular feature and follow it as it moves across the scene.

L_2 A constant rate of image acquisition would be difficult to guarantee, and probably n t be desirable in a general context. Sampling rates will be affected heavily by computational demands on the system and vehicle velocities dictated by higher level concerns.

L_3 A linear path would be an unacceptable or highly improbable trajectory in most every situation except extended flight.

L_4 *Static scenes are the least* likely — winds blow, clouds move; often a moving object in a scene is the one of most interest.

Dissatisfactions:

D_1 The analysis should proceed sequentially as the imagery is acquired. To insist that all data be available before scene measurement can be begun would eliminate one of the principal goals of the process — to provide timely spatial information for a vehicle in motion.

D_2 The EPI partitioning, through its selection of the temporal over the spatial analysis of images, could not provide *spatially coherent results*. It produced point sets. We attempted clustering operations on these, but were never satisfied with such a *post hoc* approach. The proper approach to obtaining spatial coherence in our results would begin with not losing it in the first place.

*This research was supported by DARPA Contracts MDA 903 86 C 0084 and DACA 76-85-C-0004.

2. NEW APPROACH TO EPI ANALYSIS

2.1. *Generalizations*

We have developed generalizations to our earlier approach that enable us to resolve L_1, L_2 and D_1 and D_2. Arbitrary and varying camera attitudes and velocities are permissible in our new formulation, and we process the data sequentially as acquired, forming estimates, of increasing precision, descriptive of spatial contours rather than points. The generalization also suggests a mechanism for dealing with the non-linear path question of L_3, although we have not pursued this as yet.

L_4 rises as an incompatibility between our performance desires for a vision system and our definition of the task we choose to address. We wish to build 3-dimensional descriptions of scenes, and it is inappropriate to expect this to be possible if our view of the scene is undergoing change unrelated to our active pursuit of observations. In [2] we discussed this motion issue, and suggested how to recognize its presence in a scene. Once distinguished from the static elements, it would be possible to invoke higher-order models and filters to estimate these objects' dynamics (see Broida [4] and Gennery [8]), but our current interest is in modelling static structure, and we will not pursue this issue here.

In common with our earlier work, our new approach involves the processing of a very large number of images acquired by a moving camera. The analysis is based on three constraints:

1. the camera's movement is restricted to lie along a linear path;
2. the camera's position and attitude at each imaging site are known;
3. image capture is rapid enough with respect to camera movement and scene scale to ensure that the data is, in general, temporally continuous.

Within this framework, we generalize from the traditional notion of epipolar *lines* to that of epipolar *planes* – a set of epipolar lines sharing the property of transitivity. We formulate a tracking process which exploits this property for determining the position of features in the scene. This tracking occurs on what we term the *spatiotemporal surface* – a surface defining the evolution of a set of scene features over time. Critical to visualizing this space-time approach is obtaining an understanding of the geometry of the sensing situation, and this is described in the next section.

2.2. *Geometric Considerations of Camera Path and Attitude*

The camera is modelled as a pin-hole with image plane in front of the lens (Fig. 1). For each feature P in the scene and two viewing positions V_1 and V_2, there is an *epipolar plane*, which passes through P and the line joining the two lens centers. This plane intersects the two image planes along corresponding *epipolar lines* (note that, here, intersection and projection are, in a sense, equivalent). An *epipole* is the intersection of an image plane with the line joining the lens centers. In motion analysis, an epipole is often ıeferred to as the *focus of expansion* (FOE) because the epipolar lines radiate from it. The camera moves in a straight line, and the lens centers at the various viewing positions lie along this line. Here, the FOE is the camera path. This structuring divides the scene into a pencil of planes passing through the camera path. We view this as a cylindrical coordinate system with axis the camera path, angle defined by the epipolar plane, and radius the distance of the feature from the axis. Note that a scene feature is restricted to a single epipolar plane, and any scene features at the same angle (within the discretization) share that epipolar plane. This means that, as in our earlier work, the analysis of a scene can be partitioned into a set of analyses, one for each plane, and these planes can be processed independently. In section 3 we describe how we organize the data to exploit this constraint.

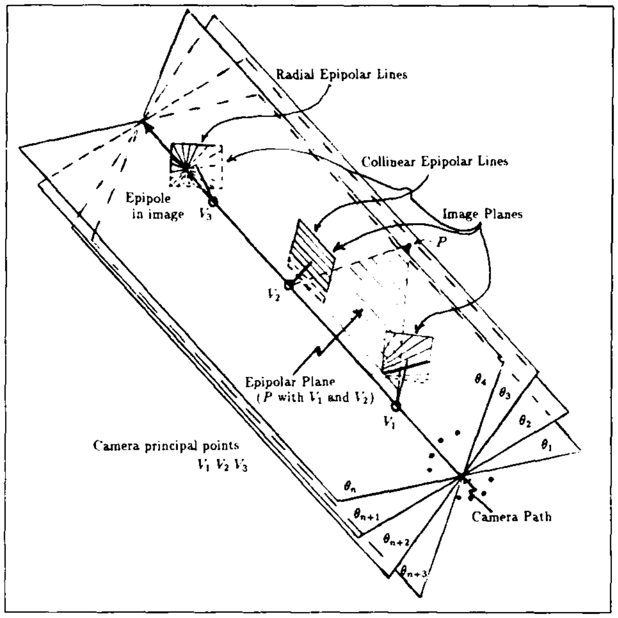


Fig. 1. General Epipolar Configuration.

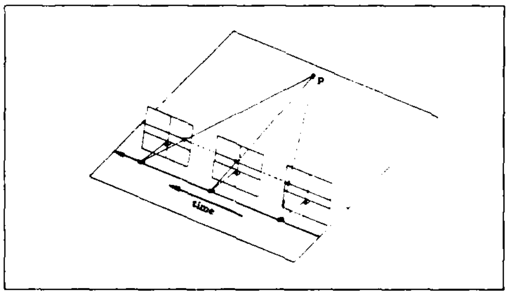


Fig. 2. Right-to-Left Motion.

Fig. 2 shows a simple motion with a camera travelling orthogonal to its direction of view. This corresponds to the situation depicted by V_2 of Fig. 1. Here, the epipolar lines for a feature such as P are horizontal scanlines, and these occur at the same vertical position (scanline) in all the images. This is the camera geometry normally chosen for computer stereo vision work. Each scanline is a projected observation of the features in an epipolar plane. The projection of P onto these epipolar lines moves to the right as the camera moves to the left. If one were to take a single epipolar line (scanline) from each of a series of images obtained with this camera geometry and compose a spatiotemporal image (with horizontal being spatial and vertical being temporal), one would see a pattern as in Fig. 3. For this type of motion feature trajectories are straight lines, as can be seen. This is the case handled by our previous analysis. If, on the other hand, the camera were moving with an attitude as shown at V_3 in Fig. 1, the set of epipolar lines would form a pattern as shown in Fig. 4. For this type of motion feature trajectories are hyperbolas. Notice that the epipolar lines are no longer scanlines – they are oriented radially and pass through the FOE. Allowing the camera to vary its attitude along the path gives rise to spatiotemporal images as shown in Fig. 5. Here, the epipolar lines form no fixed pattern, and the paths of features are neither linear nor hyperbolic – in fact they are arbitrary curves.

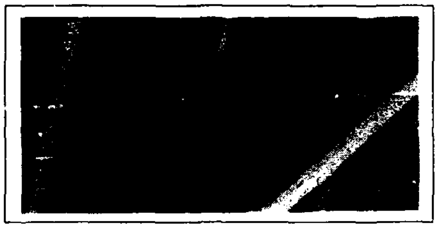


Fig. 3. Orthogonal Viewing.

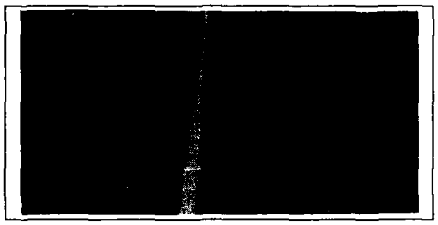


Fig. 4. Fixed, Non-orthogonal Viewing.

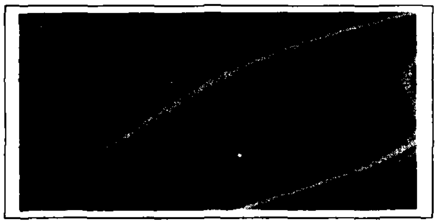
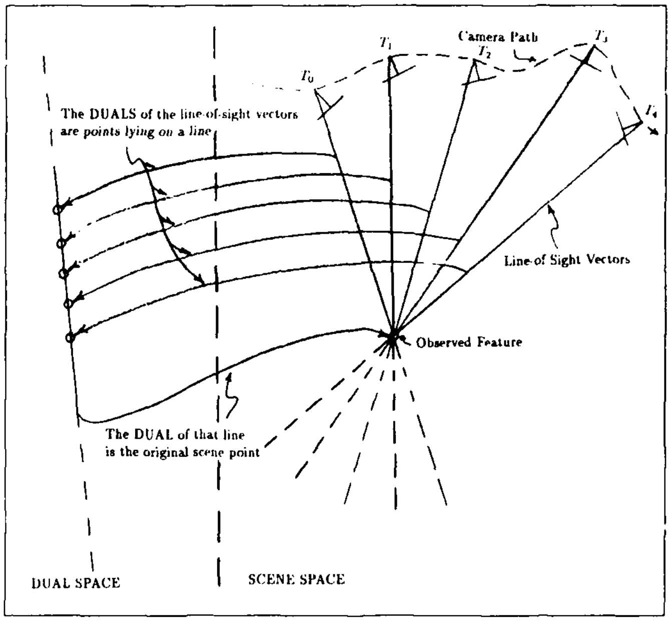


Fig. 5. View Direction Varying.

2.2. Keeping the Problem Linear

At the outset, our goal was to determine the position of observed features, and we do this by tracking their appearance in these epipolar planes. Obviously in the case of orthogonal viewing (*eg.,* as in Fig. 3 and at V_2 in Fig. 1), the tracking is linear. For general camera attitudes, including varying, it is non-linear. Computational considerations make it extremely advantageous for the tracking to be posed as a linear problem. To maintain the linearity regardless of viewing direction we find not linear feature paths in the EPIs (Figures 3 through 5), but linear paths in a *dual space.* Our insight here (see Marimont [13]) is that no matter where a camera roams about a scene, for any particular feature, the *lines of sight* from the camera principal point through that feature in space, determined by the line from the principal point through the point in the image plane where the projected feature is observed, all intersect at the feature (modulo the measurement error). The duals of these lines of sight lie along a line whose dual is the scene point (see Fig. 6): fitting a point to the lines of sight is a linear problem. This, then, gives us a metric for linear tracking of features: we map feature image coordinates to lines of sight, and use an optimal estimator to determine the point minimizing the variance from those lines of sight (we currently model only uncertainties in the image-plane observations, and not those in the sensor position).



Camera Path
T_0
T_1
T_2
T_3
T_4
The DUALS of the line-of-sight vectors are points lying on a line
Line-of Sight Vectors
Observed Feature
The DUAL of that line is the original scene point
DUAL SPACE
SCENE SPACE

Fig. 6 Line of Sight Duality.

We must, however, have a mechanism for extracting the observations of features from the individual images in which they occur, and grouping them by epipolar plane. Only in the simple case of viewing angle orthogonal to the motion is this grouping trivial (Fig. 3), and this was the case our earlier work addressed. To obtain this structuring in the general cases, we could take one of two approaches. The first is to *transform the images* from the Cartesian space in which they are sampled to an epipolar representation (see Baker [1] and Jain [12]). Because of aliasing effects and non-linearities in the mapping, we prefer to avoid this. Probably the best solution would be to use a sensor which delivers the data directly in this form (a spherical sensor having meridian scanning would accomplish this (see Gibson [9])). Such a sensor not yet being available, we choose the second approach, *transforming the features* we detect in image space to the desired epipolar space, the cylindrical coordinate system of Fig. 1. The structure we have developed for implementing this brings us several other advantages, as the next section describes.

3. THE SPATIOTEMPORAL SURFACE

3.1. Structuring the Data – Spatiotemporal Connectivity

We collect the data as a sequence of images, in fact stacking them up as they are acquired into a spatiotemporal volume as shown in Fig. 7. As each new image is obtained, we construct its *spatial* and *temporal* edge contours. These contours are three-dimensional zeros of the Laplacian of a chosen three-dimensional Gaussian (see also Buxton [5] and Heeger [11] for their use of spatiotemporal convolution over an image sequence), and the construction produces a spatiotemporal *surface* enveloping the signed *volumes* (note that in two dimensions edge contours envelop signed *regions*). The *spatial* connectivity in this structure lets us explicitly maintain object coherence between features observed on separate epipolar planes; the *temporal* connectivity gives us, as before, the tracking of features over time. See [3] for a description of how these surfaces are constructed.

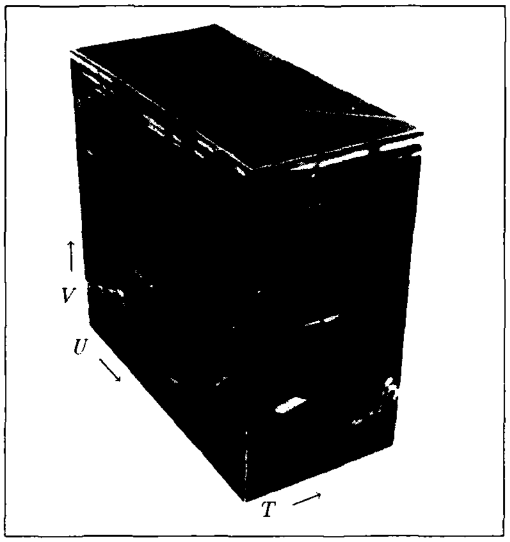


Fig. 7 Spatiotemporal Volume.

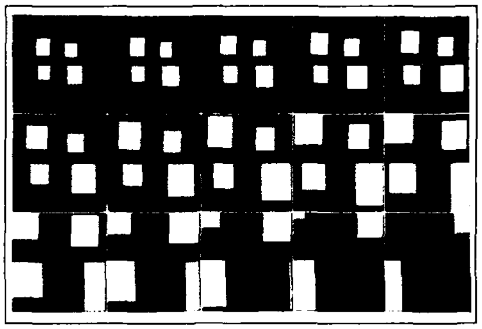


Fig. 9. Simulation: Linear Path, Motion toward Center.

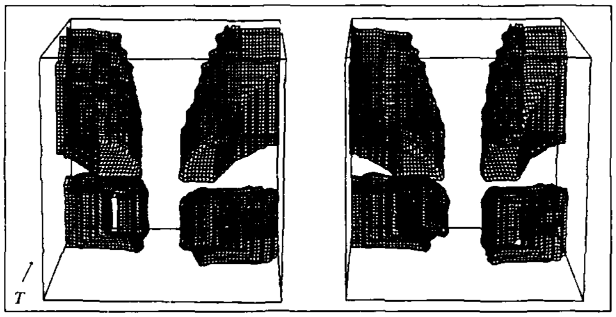


Fig 10. Spatiotemporal Surfaces from Fig. 9 Imagery.

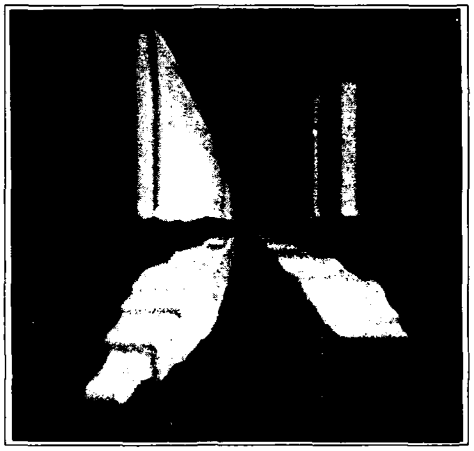


Fig. 11. Surfaces of Fig. 10 Rendered for Display.

The need for maintaining this spatial connectivity can be observed by viewing our earlier results in [2], one set of which is shown in Fig. 8. There, in processing the EPIs independently, we obtained separate planes of results. Wishing to exploit the fact that there should be some spatial coherence between these sets of points, we used proximity of the resulting estimates on adjacent planes to filter outliers. Features not within the error (covariance) ellipses of those above or below them were discarded. The remaining results (Fig. 8) were sparse and fragmented, however the problem did not lie with this filtering, but with the loss of spatial connectivity in the first place. Our separation of the data into EPIs and then subsequent independent processing of these lost the spatial connectivity apparent in the original images. We maintained instead the temporal connectivity that was critical to the feature tracking. For spatial connectivity in the scene reconstruction, spatial connectivity in the imagery must be preserved. The next three figures present a simplified example of this spatial and temporal connectivity. Fig. 9 shows a sequence of simulated images depicting a camera zooming in on a set of rectangles. Fig. 10 shows the spatiotemporal surfaces in a crossed-eye stereo wire-frame representation, and these are shown again in rendered form in Fig. 11. The spatial and temporal interpretation of these surfaces should be quite apparent.

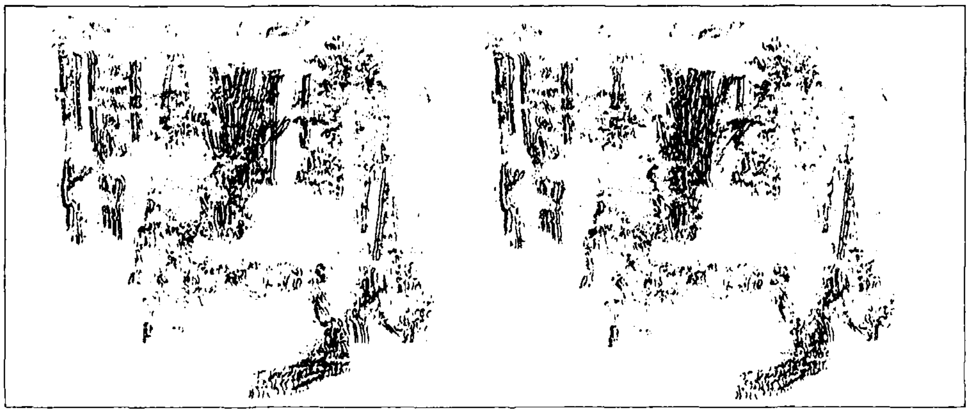


Fig. 8. Orthogonally-Viewed Scene: Results (displayed for crossed-eye viewing).

Fig. 12. Sequence 1st and 128th Images.

Fig. 13. 1st and 128th Images at $\frac{1}{8}$ Resolution.

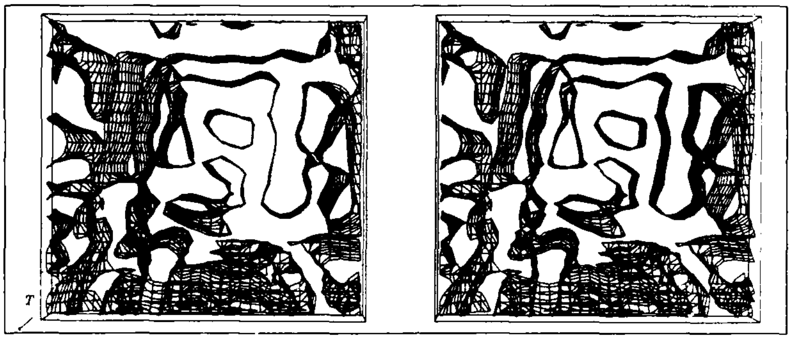


Fig. 14. Spatiotemporal Surface Representation, First 10 Frames.

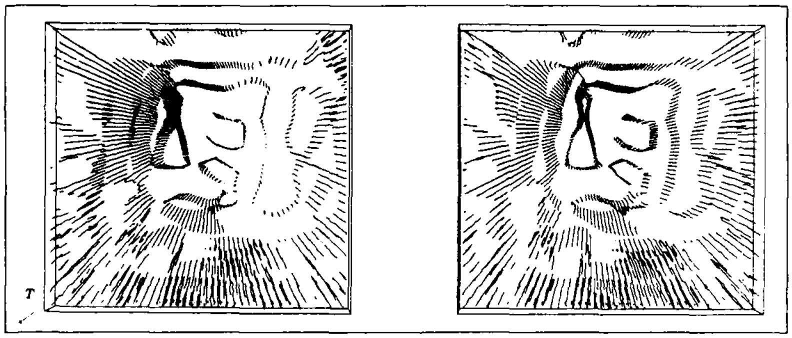


Fig. 15. Epipolar-Plane Surface Representation.

In spatiotemporal surface descriptions, feature observations bear (u, v, t) coordinates, and are spatiotemporal *voxel facets.* Fig. 14 shows a mesh description of the facets for the spatiotemporal surfaces associated with the forward-viewing sequence whose first and last images are depicted in Fig. 12. These images are much more complex than those of Fig. 9. In the interest of clarity, the surface representations we will show in the remaining figures are based on a simplified version of this imagery - one eighth the linear resolution of the originals. Fig. 13 shows these two frames at the reduced resolution.

3.2. Structuring the Data - Epipolar-Plane Representation

As mentioned in the previous section, for non-orthogonal viewing directions, epipolar lines are not distinguished by the spatial v scanline coordinate. To obtain this necessary structuring we develop within this spatiotemporal description an *embedded* representation that makes the epipolar organization explicit. Over each of the sequential images, we transform the (u, v, t) coordinates of our spatiotemporal zeros to (r, h, θ) *cylindrical coordinates* (θ indicates the epipolar-plane angle ($\theta \in [0, 2\pi]$), the quantized resolution in θ is a supplied parameter, and the transform for each image is determined by the particular camera parameters). In this new coordinate system, we build a structure similar to our earlier EPI edge contours, but dynamically organized by epipolar plane. This is done by *intersecting* the spatiotemporal surfaces with the pencil of appropriate epipolar planes[1] (as Fig. 1). We weave the epipolar connectivity through the spatiotemporal volume, following the known camera viewing direction changes. Fig. 15 shows a sampling of the spatiotemporal surfaces as they intersect the pencil of epipolar planes (every fifth plane is depicted). You will notice the obvious radial flow pattern away from the epipole (FOE). Fig. 16 shows seven of these surface/plane intersections, along with the associated bounding planes (refer to Fig. 1). The edge that all share is the camera path (*the epipole*). *Fig. 17 isolates a single surface from the top* left of Fig. 14, and shows it's spatiotemporal structure. Fig. 18 shows the same surface structured by its epipolar-plane components. Baker [3] gives details of this intersection operation on the spatiotemporal surface.

[1] Notice that, with view direction varying, the underlying epipolar plane may be far from planar in (u, v, t) space - it may undulate in a manner similar to that in which Fig. 5 varies from Fig. 3, and for similar reasons.

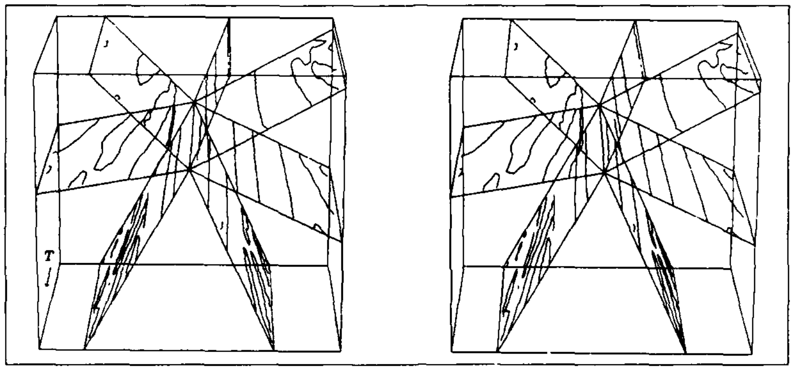


Fig. 16. Intersection of 7 Epipolar Planes with Spatiotemporal Surfaces (30 frames).

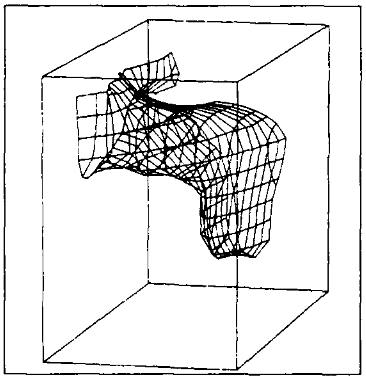


Fig. 17. Spatiotemporal Surface.

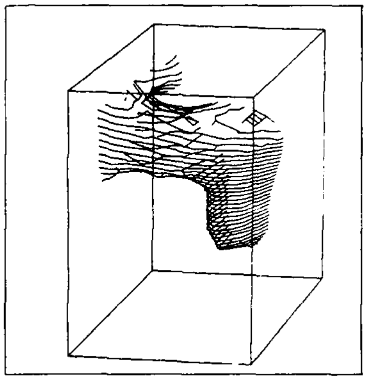


Fig. 18. Epipolar-Plane Representation.

3.3. Feature Tracking and Estimation

In Fig. 19 we show the tracking of scene features on the spatiotemporal surfaces in the vicinity of this surface, with the final pair showing, in crossed-eye stereo form, the result of the tracking after 10 frames. The coding is as follows: initiation of a feature tracking is marked by a circle; the leading observation of a feature (active front) is shown as an $\times$; lines join feature observations; 5 observations (an arbitrary number, 2 may be sufficient) must be acquired before an estimate is made of the feature's position – at that point an initial batch estimate is made, and a Kalman filter (see Gelb [7], Mikhail [14]) is turned on and associated with the feature – this initiation of a Kalman filter is coded by a square; where two observations merge, the tracking is stopped and the features are entered into the database – this is coded by a diamond[2].

As mentioned earlier, observations are expressed as line-of-sight vectors, and these are represented in the epipolar plane by the homogeneous line equation $ax + by - c = 0$. For the initial batch estimation, the coordinates ($\mathbf{X}$) of the feature are the solution of the normal equations for the weighted least squares system: $\mathbf{X} = (\mathbf{H}^\mathbf{T}\mathbf{W}\mathbf{H})^{-1}\mathbf{H}^\mathbf{T}\mathbf{W}\mathbf{C}$, where $\mathbf{H}$ is the $m \times 2$ matrix of (a_i, b_i) observations, $\mathbf{C}$ is the vector of c_i, and $\mathbf{W}$ is the diagonal matrix of observation weights, determined by the distance from the camera to the observed feature at observation position i. We estimate $\mathbf{X}$ first without weights, then compute the weighted solution and the desired covariance matrix, $\mathbf{V}$. Given a current estimate $\mathbf{X}_{i-1}$ and covariance $\mathbf{V}_{i-1}$, the Kalman filter at observation i updates these as:

$$\begin{aligned}
\mathbf{K}_i &= \mathbf{V}_{i-1}\mathbf{H}_i^\mathbf{T}/[\mathbf{H}_i\mathbf{V}_{i-1}\mathbf{H}_i^\mathbf{T} + \mathbf{w}_i] \\
\mathbf{V}_i &= [\mathbf{I} - \mathbf{K}_i\mathbf{H}_i]\mathbf{V}_{i-1} \\
\mathbf{X}_i &= \mathbf{X}_{i-1} + \mathbf{K}_i[\mathbf{c}_i - \mathbf{H}_i\mathbf{X}_{i-1}]
\end{aligned}$$

$\mathbf{K}_i$ is the 2×1 Kalman gain matrix, $\mathbf{w}_i$ is the observation weight, a scalar, based on the distance from the camera at observation position i to $\mathbf{X}_{i-1}$.

The tracking of an individual feature is depicted in Fig. 20. The camera path runs across the figure from the lower left. Lines of sight are shown from the camera path through the observation of the feature at the upper right. As the Kalman filter is begun (T_4), an estimate (marked by an $\times$) and confidence interval (the ellipse) are produced. As further observations are acquired, the estimate and confidence interval are refined. Tracking continues until either the feature is lost, or the error term begins to increase (suggesting that observations not related to the tracked feature are beginning to be included[3]). Note that although a single feature is depicted here, it is part of a spatiotemporal surface. This means that we have explicit knowledge of those other features to which this is spatially adjacent. Fig. 21 shows a contour – a connected set of features on such a surface – observed over time as it's shape evolves. Such contours are being constructed and refined over the entire image as the analysis progresses. Our current representation of scene structure is based on these evolving contours.

[2] Dickmanns [6] and Hallam [10] describe vehicle navigation controllers that similarly work sequentially, utilizing Kalman and other filters for estimating motion parameters.

[3] or the zero-crossing is erroneous, or the feature is not stationary, or the feature is a contour rather than a single point in space, or ...

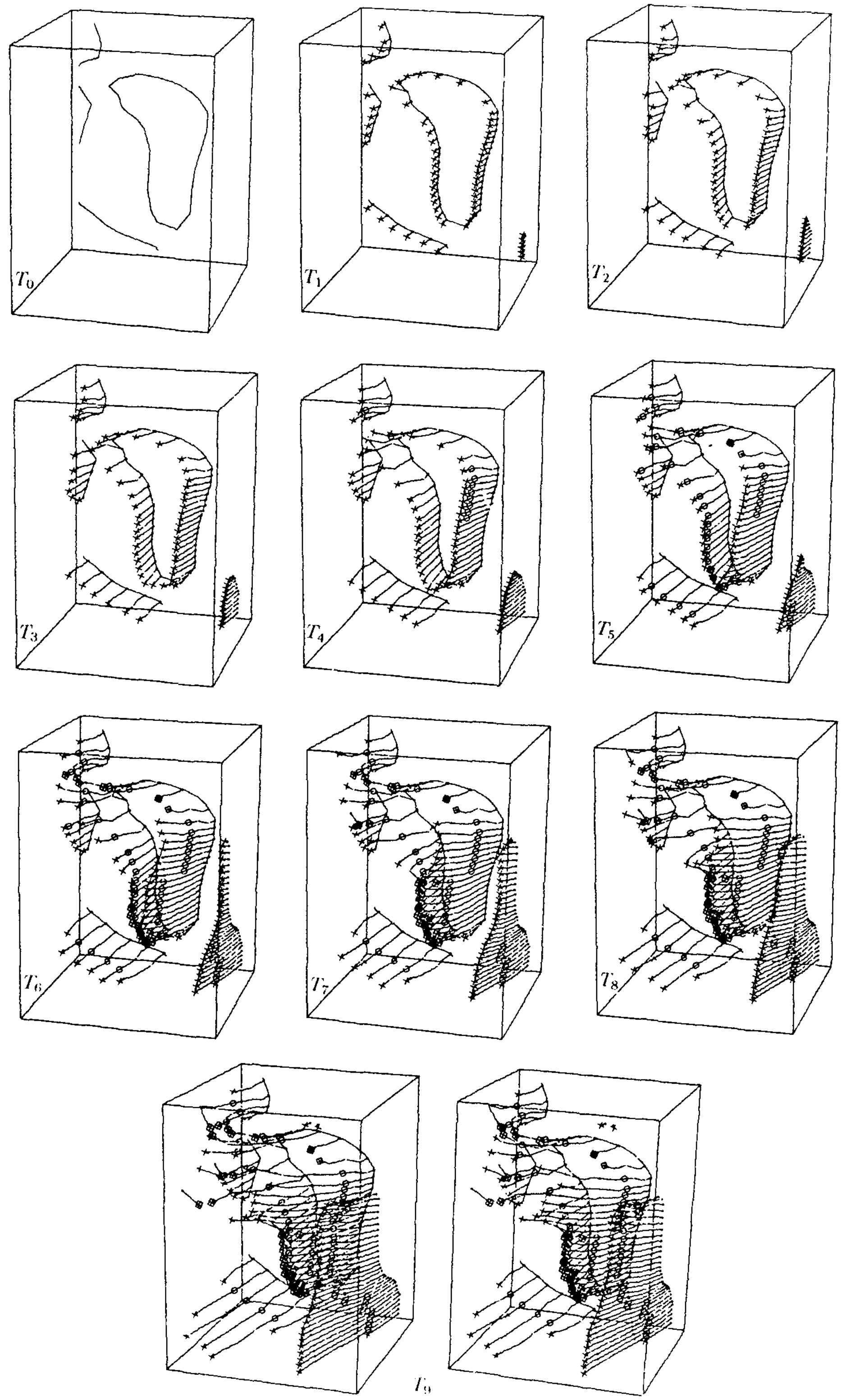


Fig. 19. Sequential Feature Tracking on the Spatiotemporal Surface.

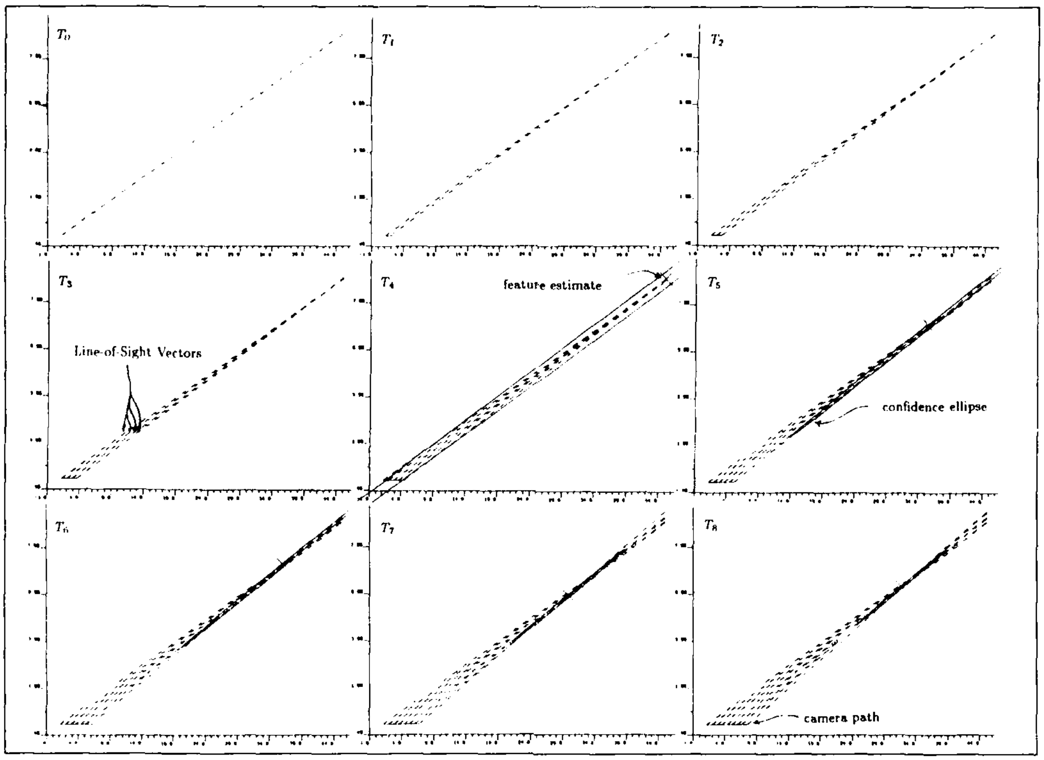


Fig. 20. Sequential Estimation.

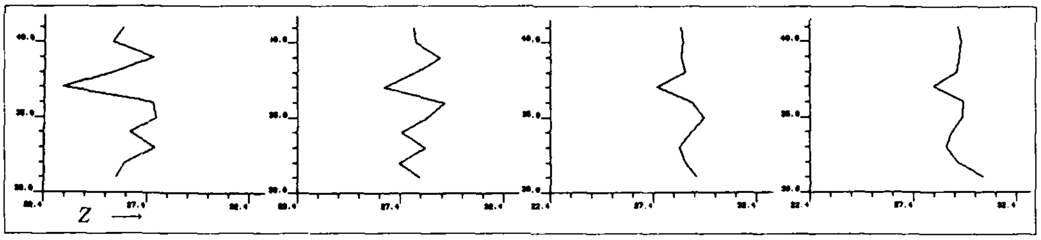


Fig. 21. Contour Evolving Over Time.

3.4. Generality from the Spatiotemporal Surface

A crucial constraint of the current epipolar-plane image analysis is that having a camera moving along a linear path enables us to divide the analysis into planes, in fact the pencil of planes of Fig. 1 passing through the camera path. With this, we are assured that a feature will be viewed in just a single one of these planes, and its motion over time will be confined to that plane. Another crucial constraint is the one we generalized from the orthogonal viewing case we know that the set of line-of-sight vectors from camera to feature over time will all intersect at that feature, and determining that feature's position is a linear problem. This latter constraint does not depend upon the linear-path constraint. In fact, the problem would remain linear even if the camera meandered in three dimensions all over the scene. This knowledge gives us a possibility of removing the restriction that the camera path be linear. All that the linear path guarantees is that the problem is divisible into epipolar planes. If we lose this constraint, then we cannot restrict our feature tracking to separate planes. The features will, however, still form linear paths in the space of line-of-sight vectors, and our spatiotemporal surface description is an appropriate representation for doing this non-planar, but still linear, tracking. The motion of features will give us *ruled* surfaces in this space of vectors, with the rules (zeros of gaussian curvature) revealing the positions of the features in the scene (visualize pick-up-sticks jammed in a box, with the sticks being the rules). This generality suggests that there is even broader application for the technique than we had initially thought. *Of course, it would be possible here to use the* pairwise epipolar constraints between images to constrain rule tracking on the spatiotemporal surface. The constraint would only apply pairwise, as the images will not have the transitivity property cited earlier. It is also worth noticing that, when the camera attitude and position parameters are not provided, the spatiotemporal surface contains everything that is necessary for determining them; but this is another problem, one which we hope to look at in the near future.

4. CONCLUSIONS

We showed, in our earlier work, the feasibility of extracting scene depth information through *Epipolar-Plane Image Analysis*. Our theory applies for any motion where the lens center moves in a straight line, with the earlier implementation covering the special case of camera sites equidistant and viewing direction orthogonal to the camera path. The generalizations obtained through spatiotemporal surface analysis bring us the advantages of:

- incremental analysis;
- unrestricted viewing direction (including direction varying along the path);
- spatial coherence in our results, providing connected surface information for scene objects, rather than point estimates structured by epipolar plane;
- the possibility of removing the restriction that fixes us to a linear path.

The current implementation, running on a Symbolics 3600, processes the spatiotemporal surfaces at a 1KHz voxel rate, with the associated intersecting, tracking, and estimation procedures bringing this down to about 150Hz, 75% of which is consumed in the surface intersection (the surface intersection would not be required if we had a sensor of the appropriate geometry). Both the feature tracking and the surface construction computations are well suited to MIMD (or perhaps SIMD) parallel implementation. With these considerations, and the process's inherent precision and robustness, spatiotemporal-surface-based epipolar-plane image analysis shows great promise for tasks in realtime autonomous navigation and mapping.

Acknowledgements

David Marimont was critical in the formulation of this work. Alex Pentland, and his Supersketch modelling and graphics system [15] were very helpful in the generating of simulated data (Fig. 9), and in rendering images of the resulting surfaces.

References

[1] H. Harlyn Baker, Thomas O. Binford, Jitendra Malik, and Jean-Fredric Meller, "Progress in Stereo Mapping," *Proceedings DARPA Image Understanding Workshop*, Arlington, Virginia, June 1983, 327–335.

[2] Robert C. Bolles, H. Harlyn Baker, and David H. Marimont, "Epipolar-Plane Image Analysis: An Approach to Determining Structure from Motion," *International Journal of Computer Vision*, Kluwer Academic Publishers, Vol.1, No.1, June 1987, 7–55.

[3] H. Harlyn Baker, "Building Surfaces of Evolution: The Weaving Wall," in these proceedings, DARPA Image Understanding Workshop, April 1988.

[4] Ted J. Broida and Rama Chellappa, "Kinematics and Structure of a Rigid Object from a Sequence of Noisy Images," *Proceedings of the Workshop on Motion: Representation and Analysis*, IEEE Computer Society, Kiawah Island, South Carolina, May 1986, 95–100.

[5] B.F. Buxton and Hilary Buxton, "Monocular Depth Perception from Optical Flow by Space Time Signal Processing," *Proceedings of the Royal Society of London, Series B*, 218 (1983), 27–47.

[6] E. D. Dickmanns and A. Zapp, "Autonomous High Speed Road Vehicle Guidance by Computer Vision," *Tenth IFAC Congress*, München, West Germany, July 1987.

[7] **Applied Optimal Estimation**, Arthur Gelb, editor, written by the Technical Staff, The Analytic Sciences Corporation, MIT Press, Cambridge, Massachusetts, 1974.

[8] Donald B. Gennery, "Tracking Known Three-Dimensional Objects," *Proceedings of the National Conference on Artificial Intelligence (AAAI 82)*, Carnegie-Mellon University, Pittsburgh, August 1982, 13–17.

[9] **The Perception of The Visual World**, James. J. Gibson, Houghton Mifflin, Boston, 1950.

[10] John Hallam, "Resolving Observer Motion by Object Tracking," *Proceedings of the Eighth International Joint Conference on Artificial Intelligence (IJCAI-83)*, Karlsruhe, West Germany, August 1983, 792–798.

[11] David J. Heeger, "Depth and Flow from Motion Energy," *Proceedings of the Fifth National Conference on Artificial Intelligence (AAAI-86)*, Philadelphia, Pennsylvania, August 1986, 657–663.

[12] R. Jain, S. L. Bartlett, and N. O'Brien, "Motion Stereo Using Ego-Motion Complex Logarthmic Mapping," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-9, No. 3., May 1987, 356–369.

[13] David H. Marimont, "Projective Duality and the Analysis of Image Sequences," *Proceedings of the Workshop on Motion: Representation and Analysis*, IEEE Computer Society, Kiawah Island, South Carolina, May 1986, 7–14.

[14] **Observations and Least Squares**, Edward M. Mikhail, with F. Ackerman, University Press of America, 1976.

[15] A. P. Pentland, "Perceptual Organization and the Representation of Natural Form," *Artificial Intelligence*, **28**, 1986, 293–331.

BUILDING SURFACES OF EVOLUTION: *THE WEAVING WALL* *

H. Harlyn Baker

Artificial Intelligence Center
SRI International
333 Ravenswood Avenue
Menlo Park, CA 94025.

Abstract

We describe a three-dimensional surface construction process designed for the analysis of image sequences. Named the *Weaving Wall*, the process operates over images as they arrive from the sensor, knitting together, along a parallel frontier, connected descriptions of images as they evolve over time. Developed to support a tracking mechanism for recovering the three-dimensional structure of a scene being traversed, other applications have since become apparent. These include rendering and computation on tomographic medical data, display of higher-dimensioned analytic functions, edge detection on the scale-space surface, and display and analysis of material fracture data. We present here displays from some of these applications. *The Weaving Wall* may be of use in representing the evolution of any two-dimensional imagery varying in a nearly continuous manner along a third dimension. We are currently looking into extending the processing to four dimensions.

1. SURFACES OF EVOLUTION

1.1. Background to the Development

Images of real objects have an inherent spatial coherence that i. general is quite obvious. This coherence persists as the objects are smoothly moved about or viewed from varying perspectives. Indeed it is often the case that clarity in the perception of a scene's structure is enhanced through either object or viewer motion. Here, the temporal coherence in the imagery reinforces that observed spatially. The research described in this paper developed from our efforts in generalizing Epipolar-Plane Image Analysis [3] to cases where both the camera and the objects in the scene are free to move about. Although we have not attained all of this yet, the process we describe here is supporting these developments.

Coupled with these viewing generalizations was a desire to enable sequential processing of the image data, all the while maintaining an evolving description of the scene. Two items in particular in this effort complicated the analysis and necessitated fairly radical redesign from our earlier system: a desire to handle sequences with varying camera attitude, and *the determination to produce spatially coherent results*. In our earlier work, we retained the temporal continuity from sequences of images and used it quite effectively in tracking and localizing features in the scene, but, in so doing, discarded the spatial continuity apparent in the individual images. Our broader goals for this research made us realize that this was information we must not lose (see [2]) our computations will *depend on both temporal and spatial image* structure. This meant that we needed to develop a mechanism for building an explicit description of the spatial and temporal evolution of images over time.

1.2. What Does It Mean For An Image To Evolve?

If objects move or a camera moves about a scene, the projections of scene features will move about in the image. If the motions are sufficiently fine, there will be no difficulty in following the various components of the scene in their independent travels. Consider, as we do for the rest of this discussion, that it is the camera which is undergoing motion while viewing a static scene. If the camera slides left at right angles to its direction of view, we would expect to see the scene slide right, with the nearer objects moving most rapidly out of view at the right. If the camera is instead looking directly along its direction of motion, we would experience a looming effect from objects lying straight ahead, and see those off to the sides slowly accelerate then fly out of sight as they approach the edges of the frame. If we had a perfect segmentation scheme[1], we might expect to be able to outline an object in the individual frames in which it is seen, and collect all of these together into a cone following the path it took as it moved about the image and then disappeared from view. This outline would change size and shape as the object came closer or receded, and as it presented different parts of its surface to view. This cone, a surface or two-dimensional manifold in a three-dimensional space, would define the evolution of the projection of that object over time. If we could track all such outlines for all objects in the scene, we would say we had a description of the evolution of the imagery over time. This is the description we want. *The outlines encode spatial coherence, and along their axes the cones maintain the objects' temporal coherence.*

Not having perfect segmentation, we will not expect to have disjoint outlines for all objects in an image. Given the variability of lighting conditions, sensor noise, reflections and texturing, etc., we *might be more realistic in expecting* nothing short of a hideous swirl of contours. But given a continuous deformation in the imagery, even this hideous swirl will sweep out a surface as we watch it in space-time. We have developed, and describe in this paper, a process which builds a representation of the spatial and temporal structure of these surfaces.

The companion paper [2] specifies the computational utility of this description in the context of our spatiotemporal tracking and surface reconstruction work. Our intention in this paper is to describe the development and operation of the surface building algorithm, to indicate applications in which it has proven to be valuable, and to suggest others in which it may.

*This research was supported by DARPA Contracts MDA 903-86-C-0084 and DACA 76-85-C-0004.

[1] This is probably a meaningless premise.

1.3. Is This Surface Building New?

Constructing surfaces in 3-D data such as this has been done before, although not in a manner that could be effective for our needs. Artzy *et al.* in [1] describe what has become the principal approach to surface reconstruction from sensed tomographic-style data; it is, in fact, the *only* surface constructor of note. In their approach, which was developed in the context of surface building from medical CT data, each surface is processed separately. The processing begins with the selection of a seed voxel of the desired density (measured in Hounsfeld units). A sequential recursive search is then used to traverse the implied surface, forming a connected set of all those facets positioned between adjacent voxels where one has density less than and the other has density greater than or equal to the chosen value. Since surface traversal is by a connected-component search through the 3-D volume, all of the data must be acquired beforehand. The surfaces traversed in this manner are guaranteed to be closed. Once one surface has been traversed, the next may be begun. This is called the ***cuberille model*** of surface reconstruction.

Although the description we seek is closely related to this, our requirements demanded a very different approach:

1. *all* the surfaces in the sequence are of interest to our tracker, not just selected ones, and we need to follow them all;
2. our aim of autonomous navigation through sequential processing means that we would never expect or desire to have all the data available in advance – our data (images from a camera) are obtained as we move, and must be processed as they are acquired;
3. a recursive traversal of individual surfaces is not appropriate, nor is processing the surfaces separately in sequence – all surfaces must be built incrementally and in parallel as each new image is acquired;
4. integral positioning of facets is insufficiently precise – for accurate mapping we need sub-pixel resolution in surface definition.
5. simple density or intensity themselves are inappropriate for surface definition – we track the evolution of image features over time, and define our features, and hence our surfaces, as being located at the zeros of Laplacians of a Gaussian;
6. we will have need of various sorts of computation on the local surface as it is being constructed, and the organization of the process must support this;

Each of our design objectives differs in important ways from [1], but the one most distinguishing of our approach is point 3. Both sequential processing and a capability for parallel implementation are crucial for a realistic tracking system.

1.4. Surface Building Operation

The surface constructor we have developed operates on images sequentially as they are acquired, knitting together a connected representation for the spatial and temporal evolution of the sequence over time. It maintains the continuity of feature paths irrespective of changes in viewing angle, or a varying rate of image acquisition; it's sole requirement of the data is that between frames, surfaces move in some direction no more than their width[2]. The processor acts as a *loom* during surface construction, with a wall of accumulators meeting each image in sequence and weaving its elements into the mesh of surfaces it has prepared behind it. From this action we give it its name, the *Weaving Wall.*

Since the principal reason for developing the *Weaving Wall* was for use in motion sequence analysis, we will develop its operation in the context of Laplacians. For additional applications which we will touch on later, other measures may be more appropriate – for example, we use Hounsfeld density when we process CT data, and track not density zeros, but zeros with respect to a bias, the chosen surface density value. But the distinction is only incidental to the development, and we will work for now with the Laplacians.

2. WEAVING WALL DESIGN AND IMPLEMENTATION

2.1. Local Surface Elements from the Volumetric Data

One characteristic presented in the derivation of the Artzy *et al* surface reconstruction process, a binary relationship on the voxels, holds for an important element of our processing. In tomographic reconstruction, voxels bear density measures, and for a chosen density a voxel is either *inside* (which includes *on*) a surface – its value is at least as great as that chosen – or *not inside* – its value is less than that chosen – and there can be no path from one to the other that does not cross the surface frontier. In our case the situation is identical: a voxel bearing a value from a Laplacian of a Gaussian is either positive or non-positive, and a path from a positive voxel to a non-positive one must pass through zero. This guarantees that our surfaces will be closed – in fact they will enclose positive or non-positive voxel values, depending on our choice. This definition ensures that each surface is a Jordan curve (Jordan surface)[3]. In essence, this means that a surface S divides R^3 into 2 volumes: that inside and that outside of S, with no holes, knots, et cetera, upsetting the continuity of the surface. This allows a finite case analysis of the local surface.

In two-dimensional images, zero-crossings will form closed *regions* composed of pixels having, say, positive Laplacian value. In three dimensions the zero-crossings will form closed *volumes* composed of voxels having Laplacian values of the same sign. In 2-D, we define the region frontier to be a contour composed of two types of edge element: a U edgel is oriented vertically and positioned between oppositely-signed horizontally-adjacent Laplacian pixels, and a V edgel is oriented horizontally and positioned between oppositely-signed vertically-adjacent Laplacian pixels (see Fig. 1). Interpolating the edgel between the two pixels allows it to be positioned up to ±0.5 units horizontally or vertically from the boundary between them. When working with an image sequence, we form a three-dimensional dataset by treating the collection of images as a set of arrays of voxels each one pixel deep. Here, we define the volume zero-crossing frontier to be a surface composed of three types of voxel facets: U and V facets separate voxels horizontally and vertically, as in the 2-D case, while T facets separate voxels temporally they occur between voxels at identical positions in adjacent images. Again, interpolation allows us to position the surface elements with sub-voxel precision.

[2] Otherwise they would be disjoint, and form two volumes, not one.

[3] In certain situations they can have multiple points, but each will have a different normal.

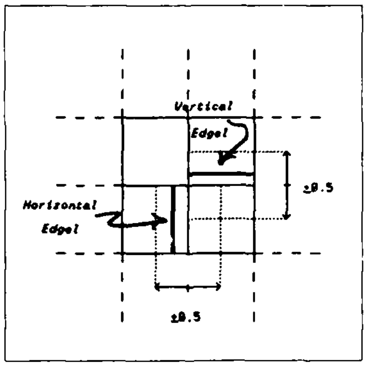


Fig. 1. Region Edgels in Two Dimensions.

Having defined what our surface elements are, we must demonstrate that we can obtain them from the data. We will do this with reference to both the 2-D and the 3-D cases, and will use both monocular and crossed-eye stereo displays to indicate the geometry behind the processing.

Given the local nature of the above definition, where facets are positioned between adjacent voxels based solely on their Laplacian values, it would seem that detecting a facet should be able to be accomplished with very simple tests applied to only local parts of the dataset. In fact, everything necessary for determining the local surface structure is available in each 2 x 2 x 2 window of the data. Consider a 2 x 2 x 2 set of voxels containing Laplacian values. There are 2^8 different combinations of positive and negative (more precisely, positive and non-positive) Laplacian values in these 8 voxels. If some of these voxels are of different sign, then we have a surface (or several surfaces) passing through this 2 x 2 x 2 part of the data set[4]. Presume that we have processed at least one image, and are now processing the next in the sequence. Our scanning is bottom to top, and within that, left to right (the development for a parallel implementation differs in detail, and has not been completed). We label our facets as indicated in the stereo display of Fig. 2, where the U facets are from the set *(U U-1 U-2 U-3)*, the V facets are from *(V V-1 V-2 V-3)*, and the T facets are from *(T T-1 T-2 T-3)*. Ignoring boundary conditions that occur at the edges of the frame, we can see that a simple 8-bit index based on the signs of these voxels can inform us of the local surface structure. Addressing voxels by their $\{uvt \mid u, v, t \in (0, 1)\}$ relative indices, we can number them from 0 through 7, with 0 being the near lower left (0, 0, 0), 1 being the far lower left (0, 0, 1), 7 being the far upper right (1, 1, 1), et cetera. If voxel 7 is positive (termed *on*), and the rest are non-positive (termed *off*), our 8-bit index is 200_8, and we will have a single surface having facets at U, V and T (Fig. 3a). The position of these facets will depend on the values of voxels 3, 5, and 6 with respect to voxel 7. The figure to the right uses circles to encode voxel state (borrowed from [6]), with filled meaning *on* and empty meaning *off*. If voxel 0 is also *on*, our index is 201_8 and we will have two surfaces, one as we had with 7 alone *on*, as above, and the other having facets at *U-3 V-3* and *T-3* (Fig. 3b). If only voxels 3, 5 and 7 are *on*, the signature is 250_8 and we will have a single surface with facets at *T-1 T T-2 V-1* and *U-1* (Fig. 3c). Sliding the 2 x 2 x 2 window one pixel in each direction will give us the adjacent local surface structure in that direction. Do this *throughout the entire volume*, and we will have a description of all the surfaces in the dataset.

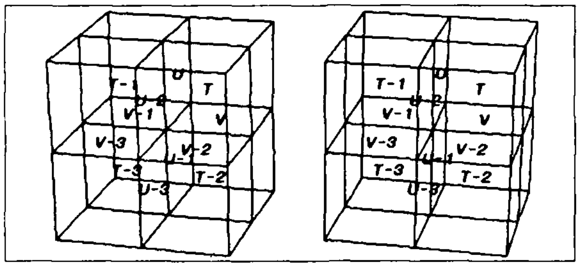


Fig. 2. Facet Labelling on 3-Space Surface.

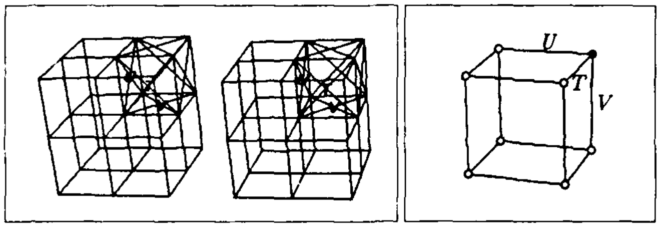


Fig. 3a. Local Surface, $n = 200_8$.

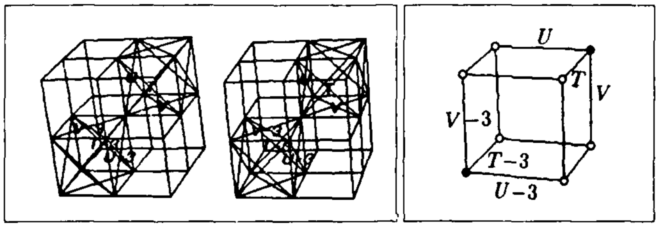


Fig. 3b. Local Surface, $n = 201_8$.

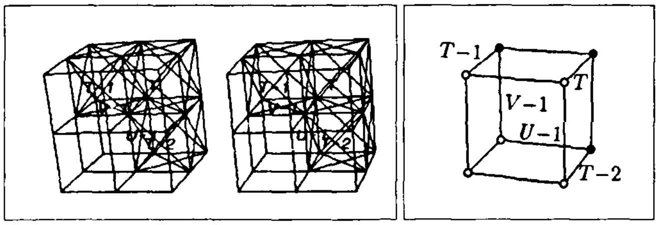


Fig. 3c. Local Surface, $n = 250_8$.

2.2. *Surface Connectivity and Facet Adjacency*

Notice in the above that we have defined neighboring voxels as being part of a particular volume if they have a face in common — edges and corners are not sufficient. This means that the complement space, made up not of enclosed voxels but *excluded voxels*, has a *connectivity definition for its voxels that includes* edges and corners. This may be best visualized in two dimensions, as demonstrated in Fig. 4. There, with pixels numbered 0 through 3, two cases are presented. First, with pixels 1 and 3 *on*, we have a single contour between pixels (0, 2) and pixels (1, 3), and it encompasses a region above that frontier (Fig. 4a). The complement region shares this contour, but encompasses the pixels below it. If, on the other hand, pixels 0 and 3 are *on*, we will have two contours passing through the 2 x 2 window, one with edgels between pixel 0 and pixels 1 and 2, the other with edgels between pixel 3 and pixels 1 and 2 (Fig. 4b). The first region encompasses pixel 0 and the other encompasses pixel 3. The two pixels *are not judged locally to be part of the same region since* they do not share a face. The complement region, however, encompasses pixels 1 and 2, in fact squeezing itself between the two

[4] This approach is an extension of a two-dimensional contour finder, having 2^4 cases, developed by Marimont.

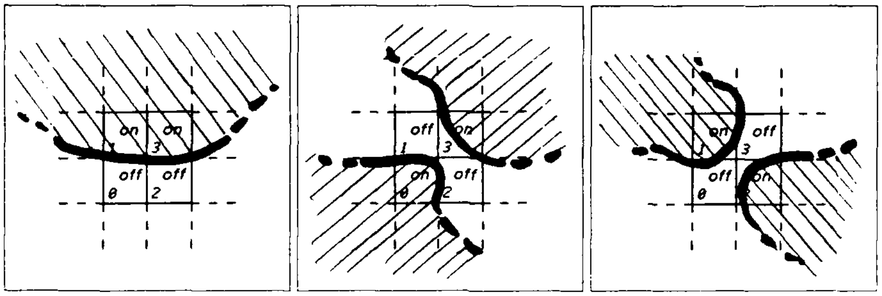


Fig. 4a. Pixels 1 and 3 *on*. Fig. 4b. Pixels 0 and 3 *on*. Fig. 4c. Pixels 1 and 2 *on*.

other regions. If a reverse *on* sense were used (negative rather than *positive*), *we would* **not** *have an identical segmentation*. Rather, pixels 1 and 2 would be parts of different regions, and pixels 0 and 3 would be part of the complement region (Fig. 4c). This means that the choice of enclosure sign affects more than just the sense of the regions; it also affects their structure.

Being consistent in this definition of adjacency is fairly crucial to maintaining a coherent surface description in 3-space. Lorensen *et al.* in [6] describe an independently developed algorithm for surface rendering based on the same use of voxel sign signatures. In their attempts to reduce the complexity of the coding, they fold the mapping about many of its symmetries. In general, *any* folding that is not undone in a later mapping would destroy the spatial coherence of our weaving. They, however, do not seek a coherent surface description, settling instead for a set of triangular facets appropriate for rendering. One of their foldings is to map the signature to its minimum with respect to number of voxels *on*. If five voxels are *on* with a signature of 37_8, this will be recoded as 340_8, having the previous five voxels now *off* and only three voxels *on*. The effect is to treat a pattern of *on/off* voxels as possibly a pattern of *off/on* voxels, complementing their true sense. The 2-D example of Fig. 4c above shows that this symmetry mapping is, in fact, not valid - it destroys the coherence of the description. In 3-D this is most disruptive at saddle points. The displays in [6] are dense enough that this isn't noticed visually.

Now, individual facets are related to those on the surface around them in the same way that individual edgels in 2-D are related to the edgels adjacent to them on a contour. 2-D edgels have two-connectivity - they have adjacent edgels to their *left* and *right* (with respect to some sense); 3-D facets have *four-connectivity* - they have adjacent facets *up*, *down*, and to their *left* and *right* (again with respect to some sense). Adjacency of facets requires a shared edge. For example, when voxel 7 alone is *on* (having a signature of 200_8), the 3 faces involved (U, V, and T) are set to indicate their neighbors: U and T are to each other's *left* and *right*, respectively, U and V are to each other's *right* and *down* respectively, and V and T are to each other's *down* and *down*, respectively. The sense is with respect to looking in at the *on* voxel from outside the face. In general, facet connectivity is of order exactly 4. Each of the 4 edges of a facet will be shared by *just one other facet*, and only when a voxel is on the boundary of the dataset will there fail to be facets to share all of its edges.

2.3. *Ancillary Surface Computations*

The coding of a local topographic signature also enables very efficient implementation of surface property computations that would otherwise be very expensive to implement, requiring total traversal of all the surfaces once completed. *First among* these is the maintaining of distinction between surfaces that are disjoint. *Each surface is given an identifying index when first* encountered, and this index is given to each face that is later found to be part of that surface. Only certain patterns of *on/off* voxels can arise from the uncovering of a new surface; ignoring boundary conditions[5], the local configuration has to be such that it includes an *on* convex corner at voxel 7. Not all such corners are actually new surfaces, however, as the local information is insufficient to indicate that the surface to which these facets belong has already been encountered. This can only be detected when the two *surfaces (the earlier and that containing the* new convex corner) come together in the 2 x 2 x 2 window. Facet linking provides for the combining of surfaces when their frontiers merge. With this approach, disjoint surface computation is primed by precompilation into the case dispatcher and finalized on the fly by the facet linker; at each stage of the processing the most concise surface description possible is the one main'ained.

Another measure that is similarly encoded in the voxel signature is the determination of surface bounding cartesian volume. As was the case with the charact·.ization of new surface arrival, the voxel configurations that contribute (or *may* contribute) to an adjustment on the enclosing cartesian frame for a surface can be characterized by their local topography. A convex corner with voxel 7 *on*, as above, may affect the lower limits in all of u v and t for the surface. *Similarly, voxels 2 and 3 on can mean an altered* upper u limit and an altered lower v limit for their surface (see Fig. 2), but cannot affect the t limits (again, ignoring boundary conditions). These tests are compiled into the case dispatcher as appropriate.

This attribute coding and incorporation into the weaving process does two things: it eliminates a later (and very expensive) sequential search; and it minimizes the time spent overall by restricting the calculations to only those parts of the surface where they are necessary. Fig. 5 sketches the local voxel configuration for a case having signature value of 267_8 (a saddle point).

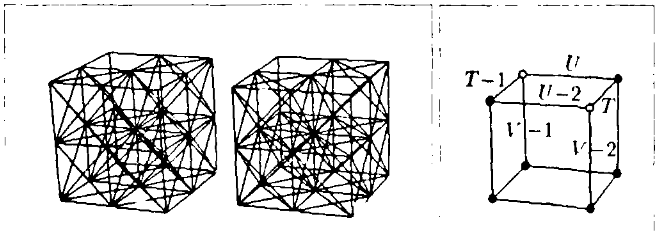


Fig. 5. Voxel Configuration, 267_8.

[5] All boundary conditions are taken care of by 1-D or 2-D versions of the 3-D operations outlined here.

The act of creating a new facet also maintains gradient mean and variance statistics for each surface. These can be of use in filtering surfaces by their significance - a low mean indicates a subtle surface more likely to be spurious.

2.4. Economies in the Construction

An advantage of the sequential processing - top to bottom and left to right - forced on us by using a sequential machine, is that we know when processing that certain facets in the $2 \times 2 \times 2$ window have already been created and linked together. In fact, at any given (u, v, t), we are only creating facets of type U, V, and T, and linking facets that are adjacent across the three edges of that octant. In Fig. 5 only two new facets and four links were need in constructing the fairly complex local surface for that case. All other facets and links were created earlier (refer to Fig. 2.).

Also important to note is that the state of voxel 0 affects neither the face and surface creation nor the face linking operations. There are no U, V or T facets in that octant, nor links to any of these facets. In effect removing one bit from our signatures, this lets us halve the state space. Exploiting this coding efficiency, the surface construction code for case 267_8 of Fig. 5 is also appropriate for the case of signature 266_8, as shown in Fig. 6. Notice that this is not a saddle point, and locally involves not one but two surfaces.

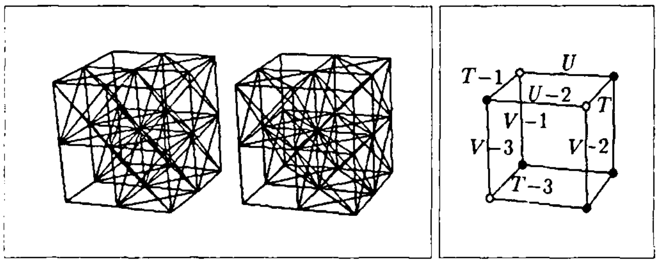


Fig. 6. Voxel Configuration, 266_8.

Fig. 7 shows a local surface configuration differing from that of Fig. 5 by, again, a single sign bit - voxel 5 here is *off* - and this gives it a signature of 227_8. This is identical topologically to the local surface of Fig. 6, but is oriented differently. This slight change leads to a rather different set of instructions for the weaver, including the creation of a new surface.

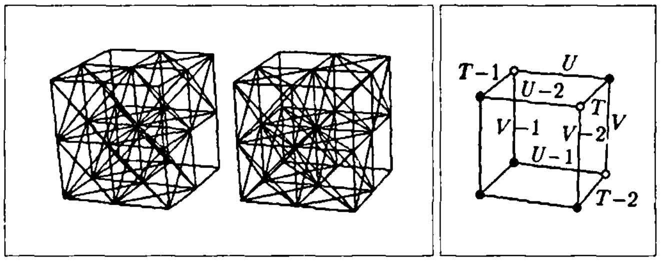


Fig. 7. Voxel Configuration, 227_8.

2.5. Representational Duality

A surface representation composed of the facets separating *on* and *off* voxels, as indicated in Fig. 2, is one way of viewing the results of the *Weaving Wall*. This is the view in the *cuberille* model, although without the interpolation (which in effect allows our facets to be of varying size). Each planar facet has integral coordinates along 2 axes, an interpolated position along the third, and an estimate of surface normal computed by the 3-D gradient convolutions. An enriched view of the surfaces is obtained if one chooses to consider them in a dual sense, the principal elements of which are the nonplanar patches formed by arcs joining adjacent facets. The shape of the patches can be estimated by interpolation of their vertex (facet) normals. Since the boundaries are oriented along the three principal axes, this gives a cartesian ruled surface representation, as indicated in Fig. 8, showing, in crossed-eye stereo form, a dart-shaped cone of circular cross-sections. Patches can have from 3 to 7 linear arcs on their boundaries: 3 is a triangle (Fig. 9 right); 4, 5 and most 6's are simple non-planar shapes (Fig. 9 right); some 6's and all 7's are saddle points (Fig. 9 left).

2.6. Locating Facets

We have discussed the structure of our surfaces and their construction from local surface elements, but have not described how we locate and parameterize those individual elements - the facets. Our data is three-dimensional, two being spatial with the third being temporal, and, in determining the facet positions and orientations, we convolve this data with a battery of three-dimensional filters[6]. Facet *orientation* is determined by computing the three partial directional derivatives about each voxel. We compute these derivatives by forming a partial derivative of a chosen 3-D Gaussian and convolving it in the appropriate di-

[6] We treat the three dimensions as isotropic.

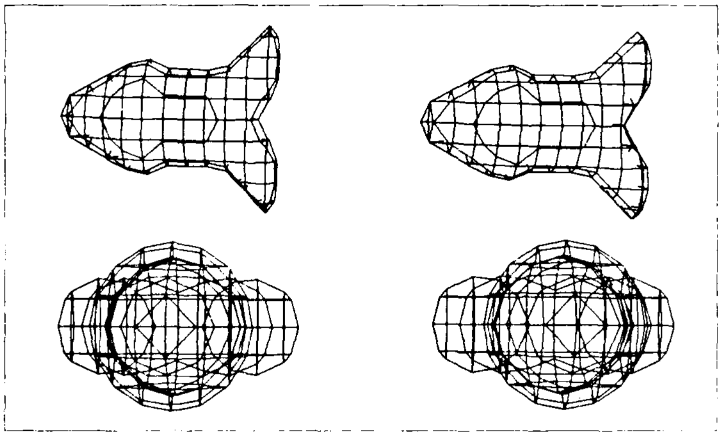


Fig. 8. Cartesian Grid Surface: top view above, front view below.

rection at each voxel. Computing facet *position* is accomplished in a similar manner: we compute the sum of the second partial directional derivatives of the Gaussian at each voxel, and position the facets by linear interpolation at zeroes of these values.

The Gaussian is a separable filter, so we can obtain the three directional derivative components as:

$$\begin{aligned}
\partial_u G(I) &= G'_u \otimes G_v \otimes G_t \otimes I \\
\partial_v G(I) &= G_u \otimes G'_v \otimes G_t \otimes I \\
\partial_t G(I) &= G_u \otimes G_v \otimes G'_t \otimes I
\end{aligned}$$

where $\otimes$ denotes convolution, G_u is a $1 \times k$ vector of Gaussian weights, G_v is a $k \times 1$ vector of Gaussian weights, and G_t is a $1 \times k$ vector of Gaussian weights, and is applied in the third dimension over a set of k images, centered at image I_t. The G's are derivatives of the appropriate Gs.

The Laplacian is also separable (Huertas [5] and Marimont (who developed this independently, and did it for higher dimensions)) so it can be computed as the sum of the second partial directional derivatives:

$$\begin{aligned}
L(G(I)) &= \partial_u^2 G(I) + \partial_v^2 G(I) + \partial_t^2 G(I) \\
&= G''_u \otimes G_v \otimes G_t \otimes I + G_u \otimes G''_v \otimes G_t \otimes I \\
&\quad + G_u \otimes G_v \otimes G''_t \otimes I
\end{aligned}$$

where the G''s are second derivatives of the appropriate Gs.

We can use separability to recompose these convolutions economically as follows:

$$\begin{aligned}
G_u(I) &= G_u \otimes I, && (1) \\
G_v(I) &= G_v \otimes I, && (2) \\
G_{uv}(I) &= G_u \otimes G_v(I) = G_v \otimes G_u(I) && (3) \\
G_{ut}(I) &= G_u \otimes G_t(I) = G_t \otimes G_u(I) && (4) \\
G_{vt}(I) &= G_v \otimes G_t(I) = G_t \otimes G_v(I) && (5) \\
\underline{\partial_u G(I)} &= G'_u \otimes G_v \otimes G_t \otimes I = \underline{G'_u \otimes G_{vt}(I)} && (6) \\
\underline{\partial_v G(I)} &= G_u \otimes G'_v \otimes G_t \otimes I = \underline{G'_v \otimes G_{ut}(I)} && (7) \\
\underline{\partial_t G(I)} &= G_u \otimes G_v \otimes G'_t \otimes I = \underline{G'_t \otimes G_{uv}(I)} && (8) \\
\partial_u^2 G(I) &= G''_u \otimes G_v \otimes G_t \otimes I = G''_u \otimes G_{vt}(I) && (9) \\
\partial_v^2 G(I) &= G_u \otimes G''_v \otimes G_t \otimes I = G''_v \otimes G_{ut}(I) && (10) \\
\partial_t^2 G(I) &= G_u \otimes G_v \otimes G''_t \otimes I = G''_t \otimes G_{uv}(I) && (11) \\
\underline{L(G(I))} &= \underline{\partial_u^2 G(I) + \partial_v^2 G(I) + \partial_t^2 G(I)}
\end{aligned}$$

giving us eleven one-dimensional convolutions (as numbered) for the full set of required gradient and Laplacian three-dimensional convolutions (those underlined).

In our experimental development we precompute these derivative images for a sequence by loading in all images at once and doing a massive, optimized set of convolutions. Another version of the processor, meant for true sequential operation, maintains a pipeline of k images, centered on the *current* frame t, carrying out the appropriate U and V convolutions on the newest image in the pipeline, and appropriate T convolutions on the current centered image t. The *Weaving Wall*, at frame t, requires the gradients for frame t and the Laplacians for both frame t and frame $t-1$.

2.7. Propagating Sequence Constraints

Returning for a moment to our sequence analysis work, we see that an indexing stategy similar to the above can be used to turn a complex operation on the spatiotemporal surface into a fairly simple operation when carried out in parallel locally at the facet level. The task is the tracking of features between frames. It is carried out locally by intersecting a pencil of planes with the evolving spatiotemporal surfaces and propagating tracker information along these paths (see [2]). Fig. 9 left shows a set of tracker paths on the local surface for the signature case 267_8 shown above. Tracker information is passed from the trailing to the leading edge of the patch along the lines drawn, each, in effect, tracking a feature through time on the spatiotemporal surface. Here, no folding of the signature is possible, and all 256 cases must be handled explicitly. Fig. 9 right shows the situation for the signature case 266_8, dealing with two surface patches.

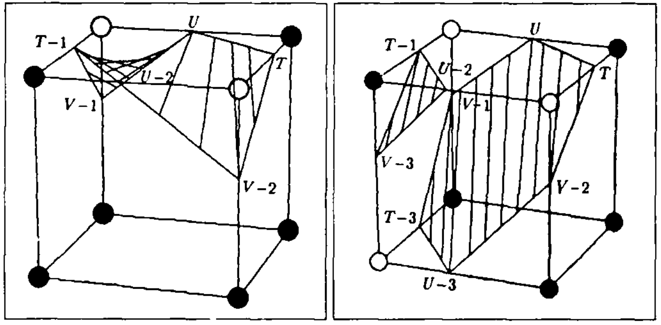


Fig. 9. Epipolar Intersections, 267_8. Epipolar Intersections, 266_8.

2.8. Rendering Surfaces

Rendering surfaces for display is generally a simpler procedure than the weaving of coherent surface descriptions, and can be seen as an adaptation of the surface patch dual representation mentioned above. For rendering, we must tesselate the patches into triangles. Given our signature mapping technique as above, this is quite straightforward, even for saddle points. Fig. 10 left shows the triangulation produced for the case of a signature of 267_8. The order of node traversal about the patches uses a right-handed rule with the thumb pointing out of the surface (refer to Fig. 2 and Fig. 5). Fig. 10 right shows the triangulation produced for a surface patch having signature of 266_8, where two surface patches are involved.

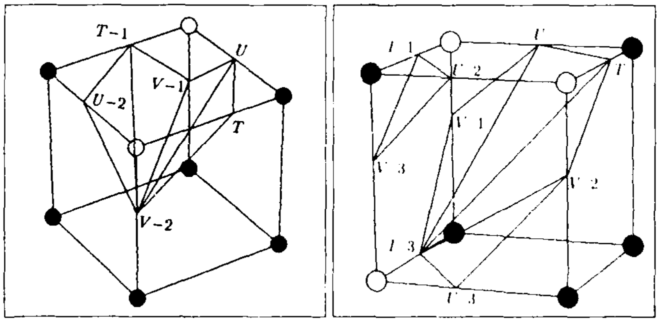


Fig. 10. Rendering Patches, 267_8. Rendering Patches, 266_8.

Actual rendering is done later, as desired, using the computed facet coordinates for spatial position, and the facet gradients as vertex normals. Triangle shading is based on bilinear interpolation of these gradients. Since disjoint surfaces are represented as distinct objects, color coding in the display and independent manipulation of the surfaces is readily attainable.

2.9. *Considerations for Parallel Implementation*

Alterations in the assumptions underlying these processes would have to be made to enable them to generate code for parallel implementations, but these would be minor rather than radical changes. The advantages of parallel implementation are quite apparent: we could divide our processing time by a large fraction of the number of pixels in an image (upwards of 65,000 for a set of 256 x 256 images). Currently the weaver operates on a Symbolics 3600 at about a 1KHz voxel rate, suggesting a parallel implementation might operate at hundreds, perhaps thousands of frames per second.

3. WEAVING WALL APPLICATIONS

Some exciting secondary applications of this research arose directly from the development of the *Weaving Wall* spatiotemporal surface-building process. This process was designed to satisfy our needs for spatially-coherent incremental tracking over an image sequence. These characteristics make it particularly useful for other applications where coherent descriptions of nearly continuous three-dimensional data is sought. Most obvious among these is the construction of surface models from CT (computed tomography) and other tomographic (*eg.*, magnetic resonance, ultrasound) medical data. We have also been applying it to visualization problems in studying the behaviour both of images with respect to spatial resolution and analytic functions of dimension higher than three.

3.1. *Medical Tomographic Data*

Although we developed it for spatiotemporal analysis, we have applied the algorithm to surface reconstruction from CT slice data - data which has no temporal component, but is totally spatial. In this we use tissue density rather than Laplacian values, maintaining the 3-D gradients for surface normal calculations. Fig. 11 shows the evolution of surfaces judged to be 'bone' (greater than 240 units) in a 70 x 30 window of a 52 image CT dataset. You can see the incremental nature of the surface development.

Fig. 12 shows another CT case study, this showing front and side stereo views of the skin, while Fig. 13 shows several stereo views (side, front, and top) of the denser bone tissue. The patient has been undergoing cranial reconstruction. These are from 46 slices of roughly 120 x 120 CT images. Interslice spacing is five times the slice resolution, which leads to a rather chunky appearance vertically. The slice sampling was increased for a section of the dataset in the area about the ear, and it would appear from the horizontal band there that the patient moved slightly as the scan adjustments were being made. There was motion also in the area of the jaw, and X-ray reflections from metal teeth fillings.

Although the surface display aspects of this technique are evidently quite worthwhile, bear in mind that the primary representation is a surface model, with all the connectivity appropriate for full model-based computation (*eg.*, finite element analysis, symmetry mappings, elastic deformation operations, et cetera).

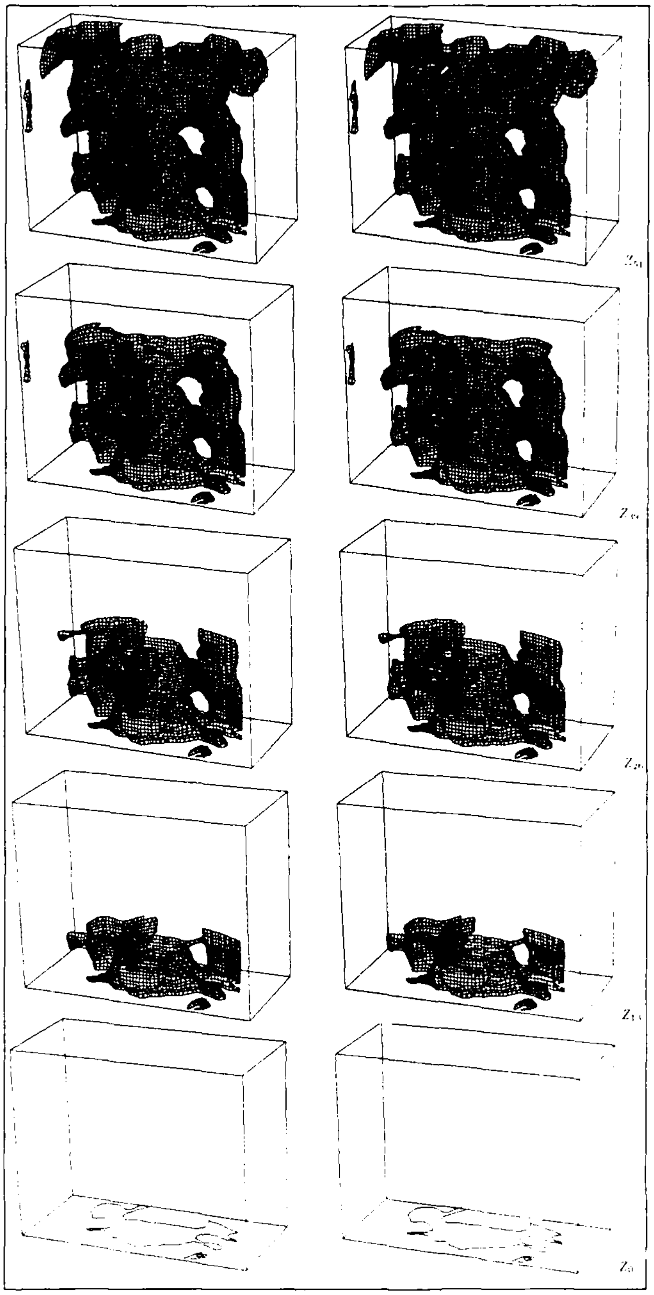


Fig. 11. Slice Evolution of Spine

3.2. *Images in Scale-Space.*

One of the standing issues in the motion sequence analysis work, and in computer vision in general, is the selection of a Gaussian to be the basis for detection operations; in effect, selecting the scale of analysis. We have done some limited experimentation with surface building where the third dimension is Gaussian scale (σ). Building a surface of the evolution of an image as its resolution varies provides a vivid picture of how Witkin's 1-D scale-space studies [8] can extend to images. Fig. 16 shows a surface constructed at the 3-D Laplacian zeros obtained by applying a battery of increasingly larger Gaussians to the image of Fig. 14. This shows eight images, each differing in σ by 0.5 from the one before. Fig. 15 left shows the zero-crossings of the smallest Gaussian, and its right shows the zero-crossings of the

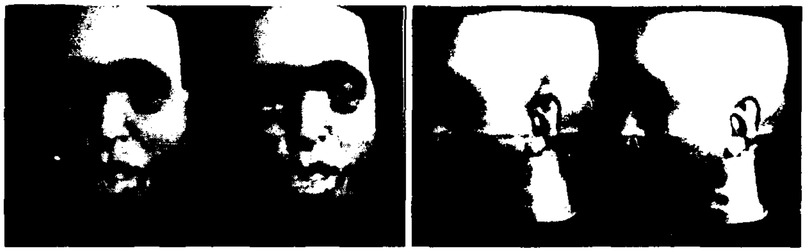
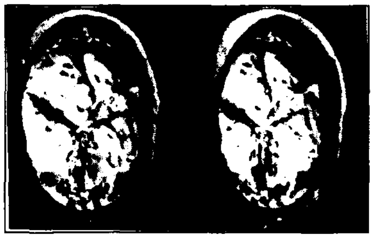


Fig. 12. CT Soft Tissue - Front. Side



Top View, Skull Interior

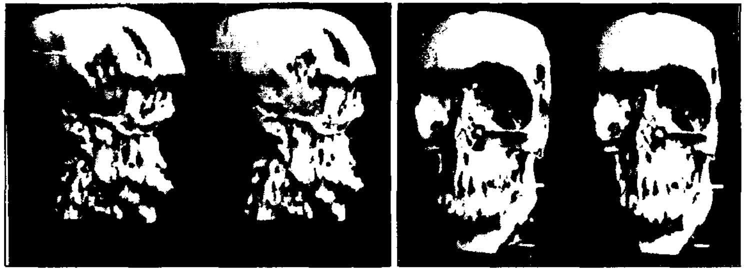


Fig. 13. CT Skull Views - Side Front

Fig. 14. First Scale-Space Image.

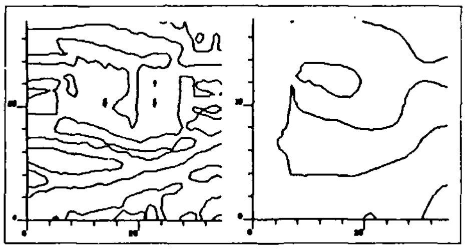


Fig. 15. 1^{st} and 8^{th} Zeros

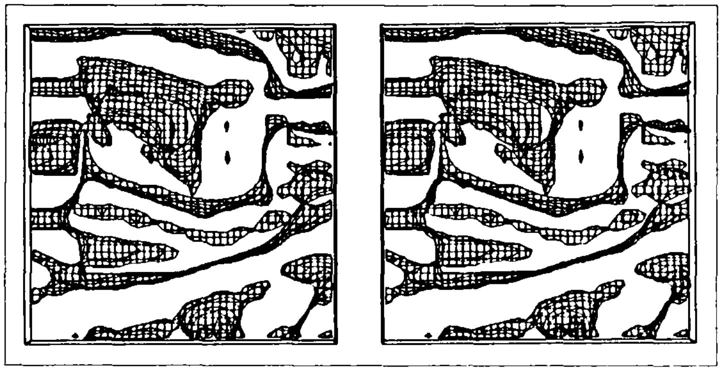


Fig. 16. Scale-Space Surface.

largest Gaussian – these are the extremes of which Fig. 16 represents the continuum. The most stable representation of a feature in this space may be at that part of its evolution exhibiting minimum spatial velocity with respect to $\Delta\sigma$. The connected scale-space surface makes this stability explicit.

In the near term we will be modifying the *Weaving Wall* to produce four-dimensional surfaces, where the first three are the spatial and temporal as before, and the fourth is Gaussian scale. Our intention is to use the most stable representation of a feature as its instantiation to be tracked. The linear estimators will then use these more appropriate σ values in determining observation weights and in estimating the resulting spatial precisions. Tracking will be occurring at all scales at once. In other applications, such as the processing of magnetic resonance images, we will be attempting to use the stability of a contour in this scale dimension as a measure to combine with gradient strength in estimating both contour scale and significance.

Another motivation for four-dimensional surface reconstruction, still lying ahead in our work, is its applicability to describing the evolution of three-dimensional (rather than two-dimensional) events over time. Rather than having a fourth dimension being *scale*, we could make it *time*, as we have in our tracking work. Our first motivating force in this is a collaborative effort we are beginning in representing the time-dependent geometry of materials as they fracture. An equally exciting application lies in capturing the temporal relationships in 3-D tomographic data – building dynamic models of, for example, beating hearts or rotating vertebrae.

3.3. Analytic Functions

The Weaving Wall has proven to be a useful tool in object representation studies. Our research group studies representation issues for object modelling, and has developed a modelling facility based on Superquadrics [7]. Abstracting from this representation, Hanson [4] has developed a higher-dimensioned hyperquadric formulation, a three-dimensional projection of which gives a superset of the superquadric primitives. Experimenting with these higher dimensioned objects is complicated by the impossibility of viewing in these higher dimensions. To facilitate this, we display sequences of three-dimensional projections of these n-dimensional objects ($n > 3$), and use the *Weaving Wall* in this rendering. Fig. 17 shows a sequence of such projections through a four-dimensional surface. Viewing such frames as a sequence gives us insight into a surface's four-dimensional structure.

Other applications of the surface-building process described here include representation of surfaces from other two-dimensional sensing domains (*eg.*, ultra-sound, geology), display of surface fracture reconstructions (2-D over time), and the colorization of black and white film. We should have preliminary results from the first two areas shortly. In general, this can be used in any application where a depiction or computational representation is desired of the evolution of a two-dimensional pattern varying in a continuous manner over a third dimension, be that dimension time, space, viewing position, resolution, or whatever.

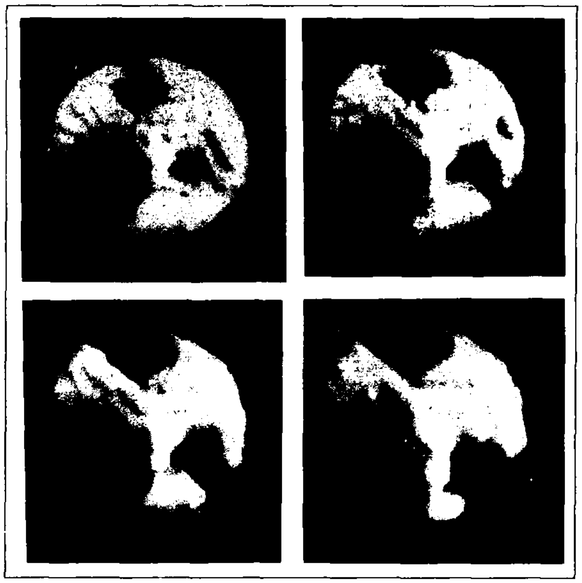


Fig. 17. Four 3-D Projections through 4-D Surface.

4. CONCLUSIONS

The *Weaving Wall* was a necessary development for our continuing research in sequence analysis. The structures it produces provide for our current needs in tracking features on the spatiotemporal surface, and seem also to be most appropriate for camera solving and for our intended extension of the analysis to nonlinear camera paths. The form of its implementation has enabled us to implement many complex geometric three-dimensional image operations at a very local level, and to maintain important surface properties at minimal cost. The applications to medical imaging, display of higher-dimensioned analytic functions, scale-space computations, et cetera, suggest that we have captured, in our surface building process, a powerful and general tool. We will be developing it further in these and other directions, and expect we will find similar variety in application for its higher dimensioned counterparts.

Acknowledgements

Bob Bolles and David Marimont were critical in the development of this work, with Dave providing insights for both the geometry and mathematics of the tracking process, the design of the surface builder, and the approach taken to higher dimensioned convolution. Alex Pentland, and his Supersketch modelling and graphics system were very useful in producing rendered displays for the surfaces presented here. Andy Hanson provided the data and motivation for the hyperquadric analytic surface depictions. The medical CT data is courtesy of Dr. C. Cuttings, New York University, and CEMAX Corporation, Santa Clara, California.

References

[1] E. Artzy, G. Frieder, G. T. Herman, "The Theory, Design, Implementation, and Evaluation of a Three-Dimensional Surface Detection Algorithm," *Computer Graphics and Image Processing*, 1981, 15, 1-24.

[2] H. H. Baker and R. C. Bolles, "Generalizing Epipolar-Plane Image Analysis on the Spatiotemporal Surface", in these proceedings, DARPA Image Understanding Workshop, April 1988.

[3] R. C. Bolles, H. H. Baker, and D. H. Marimont, "Epipolar-Plane Image Analysis: An Approach to Determining Structure from Motion," *International Journal of Computer Vision*, Kluwer Academic Publishers, Vol.1, No.1, June 1987, 7-55.

[4] A. J. Hanson, "Hyperquadrics: Smoothly Deformable Shapes with Convex Polyhedral Bounds," accepted for publication in *Computer Vision, Graphics, and Image Processing.*

[5] A. Huertas and G. Medioni, "Detection of Intensity Changes with Subpixel Accuracy Using Laplacian-Gaussian Masks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-8, No. 5., September 1986, 651-664.

[6] W. E. Lorensen and H. E. Cline, "Marching Cubes: A High Resolution 3D Surface Construction Algorithm," *Computer Graphics*, Vol.21, No.4, July 1987, 163-168.

[7] A. P. Pentland, "Perceptual Organization and the Representation of Natural Form," *Artificial Intelligence*, **28**. 1986, 293-331.

[8] A. P. Witkin, "Scale Space Filtering," *Proceedings of the Eighth International Joint Conference on Artificial Intelligence (IJCAI-83)*, Karlsruhe, West Germany, August 1983, 1019-1021.

Generation of Face-Edge-Vertex Models Directly from Images *

C. I. Connolly
J. R. Stenstrom

Aule-Tek, Inc., consultants to General Electric Corp. Research and Development Center

Abstract. Computer models of objects are necessary for a variety of tasks including model-based vision. Techniques are presented to construct face-edge-vertex models directly from intensity images. A corresponding technique is developed to build face-edge-vertex models from range images. The results are compared.

1. Introduction

With the recent increased interest in model-based vision[6,8,5,13] has appeared a concomitant interest in automated model creation.[1,7,10] There is also considerable interest in acquiring models to be used for mission rehearsal, robotics, and simulation purposes. This paper describes automatic procedures for building face-edge-vertex models directly from image data. Techniques are shown to be applicable and effective both to intensity and to range data.

Computer models must be geometrically accurate. Relative positions of surfaces on the model should not be very different from those of the actual object. The models must also be topologically sound, representing the object by a single orientable surface. If the models obtained from individual views are indeed sound and the view models may be placed in a common coordinate system, a boolean intersection of the view models provides an integrated multiple-view model. This is the essential technique for building models from both range and intensity image data.

2. Models from Intensity Data

2.1 Overview of Technique

Face-Edge-Vertex models are here derived directly from one or more ordinary intensity images. Objects or the individual faces of the object are represented by (closed) boundanes or *1-cycles* in the viewplane. Edges are detected in the images with the Canny[2] edge detector coupled to an adaptive threshold zero-crossing following procedure that is effective in actually finding these 1-cycles. Large dangling edge chains are connected to nearby edges when they are quite close. Edge lines are fit to approximate the contour followed by the 1-cycles.

The parameters of the imaging camera are used to generate bounding volumes for the surfaces which gave rise to the features in each image. Assuming the pinhole model i[illegible] properties are the orientation, focal point position an[illegible] length of the camera. Using this information th[illegible] *[illegible]les* comprising the features in the image can be [illegible]ed to form volumes (*2-cycles* or *Blocks*).

View volumes are intersected with volumes obtained from other views to obtain the potentially solid blocks corresponding to objects in the scene. This intersection process usually produces an ambiguous set of blocks. Each block represents a bound on the position of any surface which might lie within that volume. By generating repeated views, view solids, and intersections, solid models may be generated which closely approximate the object being viewed.

2.2 Relation to Previous Research

The approach described here is related to the work of Chien and Aggarwal.[3] The chief difference, however, is that the method described here generates Face-Edge-Vertex models from multiple edge cycles in one or more intensity images. Chien and Aggarwal require that an image be segmented into object and background. This produces silhouettes which are first converted into quadtrees representing orthogonal views and then assimilated into octrees. Chien and Aggarwal suggest that, in principle, their algorithm extends to non-orthogonal views. While the octree may be completed from non-orthogonal views, the procedure is more complicated, involving ray-tracing techniques.[4] Chien and Aggarwal point out that a large class of objects are well-modelled by the process. In particular, objects with curved surfaces are represented by a faceted polyhedral model.

One disadvantage of the octree model, however, is that any planar surface which is not parallel to one of the octree axes will *also* be broken up into many orthogonal facets -- all lying within a small region of the actual surface plane. The Face-Edge-Vertex model used here shows

* This work was supported in part by the DARPA Strategic Computing Vision Program in conjunction with the Army Engineer Topographic Laboratories under Contract No. DACA76-86-C-0007.

no a-priori preference to any view directions. The algorithm is the same whether views are orthogonal or occur at random known angles. Visible planar surfaces will be resolved into either one planar face or a small number of planar surfaces that are nearly coplanar.

In the method described here, multiple volumes are created to correspond to the multiple edge cycles in the image. These volumes must then be disambiguated into solid or empty volumes. Whether the volumes are solid or empty can often be determined from multiple views of the same scene from different directions. There are other techniques that may be useful, as will be seen later.

This work is also somewhat related to the work of Baker,[1] and especially that of Wesley and Markowsky.[12] The Markowsky and Wesley projection fleshing algorithm solves a similar model construction problem. The Wesley-Markowsky algorithm is designed to produce solid models from wire frame projections. The solid model is constructed by relying on an existing wire frame fleshing algorithm.[9] Unfortunately, this algorithm will succeed only for a complete wire frame. The algorithm is well-suited to multiple orthographic views of wire frames, with no hidden line removal allowed. When these conditions are not met, significantly more views will be necessary to produce the vertex correspondences required to complete the wireframe. While such conditions are relatively easy to achieve in a CAD-CAM environment (the intended arena for the algorithm), a different approach must be used when dealing with image data. There are several other problems in applying the Markowsky-Wesley algorithm to the construction of solid models from intensity images. In the projection fleshing algorithm, the geometry of the input data is assumed to be exact. That is, the correspondence between vertices from view to view must be exact. This will almost never be the case for actual image data. Also, Markowsky and Wesley discuss the topological aspects of their algorithm as it relates to curved surfaces, but do not provide any comparable algorithm which will behave suitably with curved surfaces. The algorithm described in this paper produces a solid polyhedral model from each intensity image, even if the object has curved surfaces. Approximating curved edges in the image by line segments will merely result in an approximate model -- rather than no model at all. We show that repeated views of an object iteratively produces a faceted model approximating curved surface.

2.3 Generation of View Volumes

Volumes are generated using the following procedure: First, an image is Gaussian smoothed and processed with a suitable edge detector (e.g. Canny[2]). Then, the edges are converted to 1-chains (sets of edges) by selecting junctions and points of high curvature (when the edges are viewed as curves) to serve as vertices. It is often the case that the result of applying an edge detector to an intensity image produces gaps in what would otherwise be closed curves consisting of edges. The Canny process has been modified to attempt to follow weaker zero-crossings in the vicinity of several edge events. Gaps are produced by regions of low contrast, artifacts of lighting and imaging. To counteract this effect, the edges are extended a certain small length so that they may intersect other, nearby edges. The resulting complex of edges is now pruned so that only 1-cycles remain. Each 1-cycle is then extruded to form a 2-cycle. Each edge of the 1-cycle is used to form a back-projected face. The front and rear faces of the 2-cycle are simply planes set at arbitrary forward and rear distances from the camera. The measured camera parameters (e.g., focal length) are used to determine the shape of the 2-cycle by using the inverse perspective function. The perspective function maps points in $E^3(x,y,z)$ to points on the viewplane (X,Y) as follows:

$$X=\frac{fx}{f-z}$$

$$Y=\frac{fy}{f-z}$$

where f is the camera focal length. Differentiating x and y with respect to z yields:

$$\frac{dx}{dz}=\frac{X}{f}$$

$$\frac{dy}{dz}=\frac{Y}{f}$$

This, along with the location of the point in the viewplane, completely specifies the ray on which the actual point in E^3 corresponding to a viewplane point must lie. Two values of z are chosen (z_0,z_1) so that they fully enclose the scene volume in the z direction. These z values fix the positions of the front and rear faces of the 2-cycles. The 2-cycles are finally transformed into a common coordinate system and intersected with each other to get approximations to the objects found in the scene.

2.4 Experimental Results

Figures 1 shows one intensity view used to create a model of a jeep. Image edges automatically extracted by the modified Canny procedure are back projected to form the view solid of the bottom row. Figure 2 illustrates the second view. The view solids are intersected and the result is shown as a rendered image in figure 3. The model is deceptively simple, since it was generated from two views 90 degrees apart. Note that each wheel and the inside of the jeep are separate bounding volumes whose states (empty or full) must be determined. Here the blocks were disambiguated by the operator.

3. Models from Range Data

3.1 Previous techniques

Previous work[10] has demonstrated the feasibility of using three dimensional edges developed from multiple range views to produce wire-frame representations for an object. The Wesley-Markowsky wire frame fleshing algorithm[9] becomes directly applicable to such a wire frame. The separate view solids thus produced are transformed into a common coordinate system and intersected.

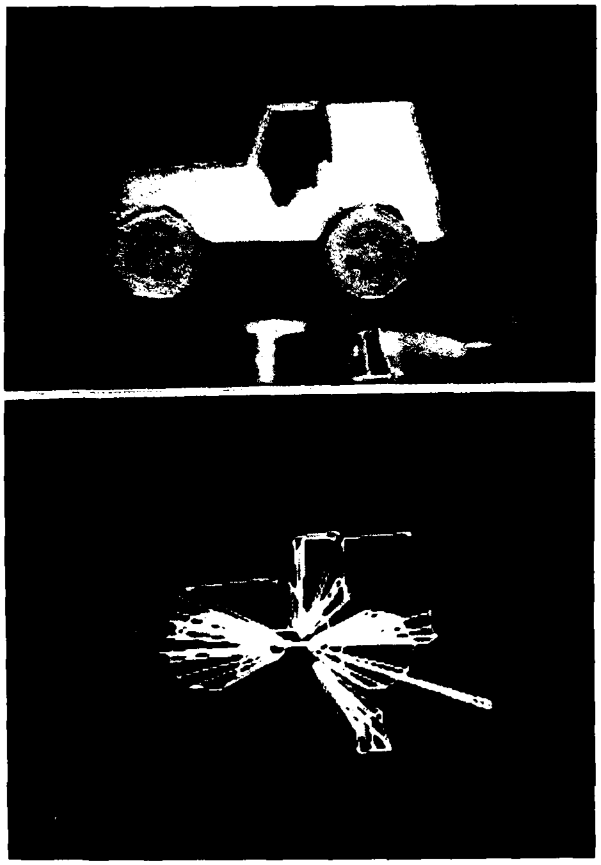


Figure 1. First view of a jeep. The intensity image and the view solid automatically obtained.

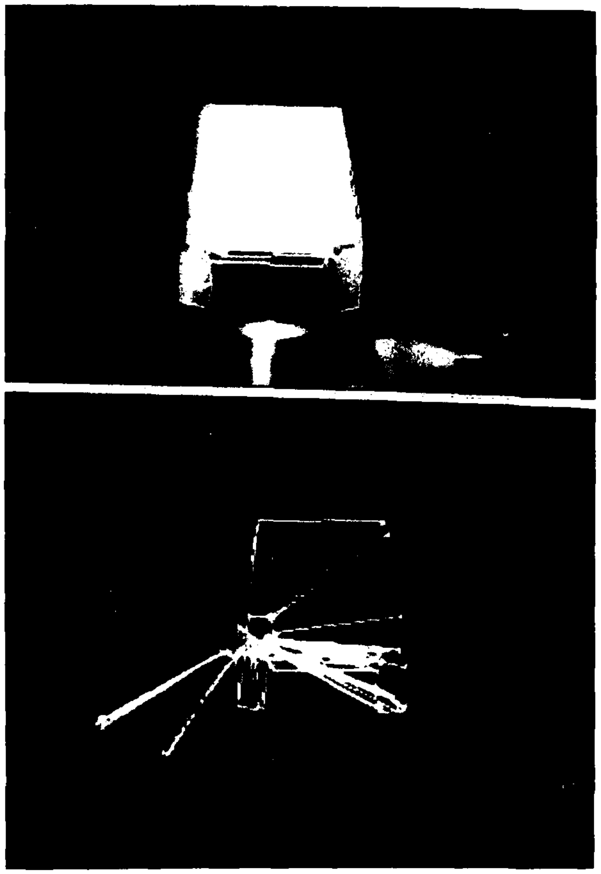


Figure 2. Second view of a jeep. Intensity image and view solid.

There are several problems that make this procedure difficult to apply. The edges that bound a visible face are connected in the range image. The casting of each intersection vertex into three dimensions guarantees preserving any 1-cycles in three dimensions. Unfortunately, there can be absolutely no guarantee of of the planarity of such edge cycles. The Markowsky-Wesley algorithm identifies possible 2-cycles. Usually, the solids which are so generated are ambiguous, and a tree of possible solid/empty labels of the volumes is the output of the algorithm. While the ambiguity cannot be completely resolved when fleshing wire-frames from engineering drawings, the ambiguity between empty and solid is absent in range data.

The new procedure avoids the planarity issue entirely by extracting least mean-squared error faces directly from the range image and wrapping auxiliary faces to close 2-cycles. In this process, the range image disambiguates space into solid and empty. The model resulting from the new procedure has strictly planar faces, and is a closed surface.

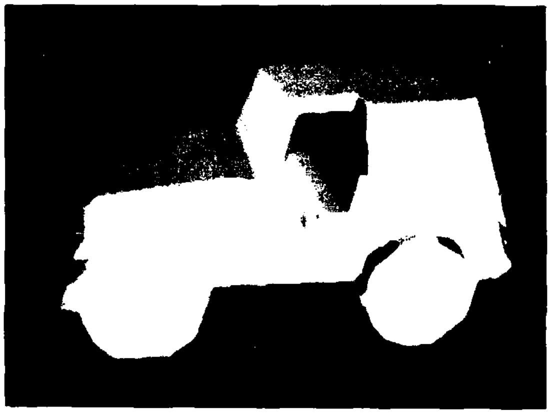


Figure 3. Rendered jeep model obtained from the intersection of the two view solids derived from intensity images.

3.2 The Algorithm

Edge cycles corresponding to significant object features are formed in range images. Edges are of two types, curvature and occlusion. Occlusion edges are represented by step edges in a range image. The edge detection techniques typically used with intensity images also provide very good results with range images. The curvature edges are problematic. Ideally these edges should follow lines of high curvature representing the boundaries of adjacent object faces. In practice these edges are difficult to extract. We have experimented with automatic extraction of curvature edges for the purposes of model creation[11] but the results, to date, have been disappointing. Although we are able to obtain reliable edge boundaries for the intensity images automatically, for the purposes of this paper, edges in range data are provided by the operator. Work continues on improving automatic techniques for extracting curvature edges.

Selected edges form large complete cycles corresponding to major deviations from planarity. A single image is assumed to contain one or more objects fully surrounded by a background surface, identifiable by its depth. The outer bounding cycle of that background surface properly contains every other viewplane cycle and is designated as the rear face. Viewplane cycles are converted to three-dimensional faces or holes (if they are coplanar with the rear face). Pixels within the current face but not in a hole are sampled. A least-squares surface fit determines the geometric parameters of the three-dimensional face within the outer edges. Sensor geometry provides placement of the bounding edges to a plane in space. The intersection of the face equation and the plane determined by the edge provides a very accurate placeme:t of the edge in three-dimensional space.

The "holes" in the outer bounding cycle (or rear face) are precisely the occlusion edges of the sensed objects. These may be outermost edges or edges surrounding holes in the sensed objects. To produce a model with holes that properly correspond to those of the sensed object, the cycles surrounding holes in the sensed object must be identified. The faces that correspond to the cycles that bound object holes are identified and collected. At this point all of the three-dimensional faces that correspond to viewplane cycles are fully formed. It remains only to wrap back-projected faces around the object and to close the rear surface.

The first step in the back projection considers each vertex in the range image cycle complex. A line is formed in the view direction that includes the vertex. Cycle, edge, and vertex links permit the identification of all the range-image cycles that include the range vertex in question. Thus, the three-dimensional faces that are encountered along the view direction from the vertex may be readily identified. The intersections of this view direction line and the identified faces are the vertices that will correspond to the range vertex. Each three-dimensional vertex is also paired and linked with the appropriate face. These corresponding three-dimensional vertices are connected pairwise, with three-dimensional edges.

The back projections are done on an edge by edge basis. Each range edge is in precisely two viewplane cycles and thus represents the boundary of exactly two three-dimensional faces. Ordinarily the same face is front-most at each end of the three-dimensional edge. In this case, forming the back projection is quite easy. The edge in the viewplane together with the view direction determine a plane which may be intersected with the forward and rearward faces, producing two edges. There are four edges for each back-projected face: these two three-dimensional edges corresponding to the range image edge, and two additional edges extending back along the line of sight, created in the first step of this section. There is another case for back-projected faces. The same face occasionally will not be front-most at both ends of a range edge. When it is not, the back projected surface must be split into two triangles meeting at a common three-dimensional vertex. The three-dimensional vertices in the front and rear face for each endpoint of the range edge, together with this common vertex, form the three corners of each triangular back-projected face.

Whether by the simple case or by the two triangle case, *every* back projected face has now been constructed. The final step constructs a valid rear face for the object model. Recall that an outer bounding rear face was created early in the process. During the rest of the procedure, faces may have been created which were coplanar with the rear face. These rear face cycles are arranged in a containment tree. The outer rear cycle, at the top level, is made up of the outer-most occlusion edges of the object and is labeled solid. Each interior cycle/face is labelled in the opposite sense of its ancestor. The resulting tree will represent the final face for the model.

3.3 Experimental Results

Polyhedral models from multiple views are intersected to refine the model. Figure 4 shows our range camera. The mirrors at the left side direct a stripe of light on an object as shown in figure 5. Figure 6 shows a range image for a single view of a jeep, the edges identified in the view, and a view solid produced by the algorithm. Figure 7 shows a second view and associated edges. Figure 8 shows the intersection of the two view solids. It is apparent that the intersection result closely approximates the original object.

4. Conclusions

The practical use of geometric and topological properties of image data to construct three-dimensional models has been demonstrated. The essential information consists both of local, geometric information obtained from step and curvature edges found in the image, and the more global, topological information inherent in the data (the cycles which define surface boundaries). Preserving this information at every step of the model creation process,

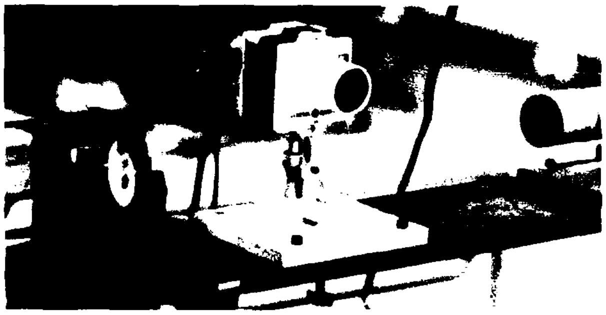


Figure 4. Range camera.

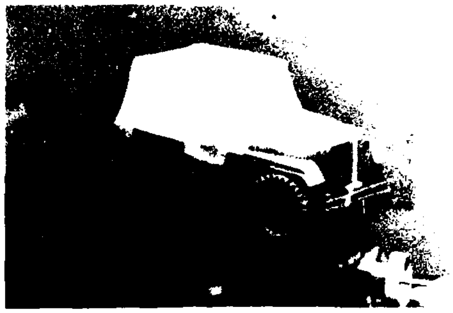


Figure 5. Light stripe on jeep.

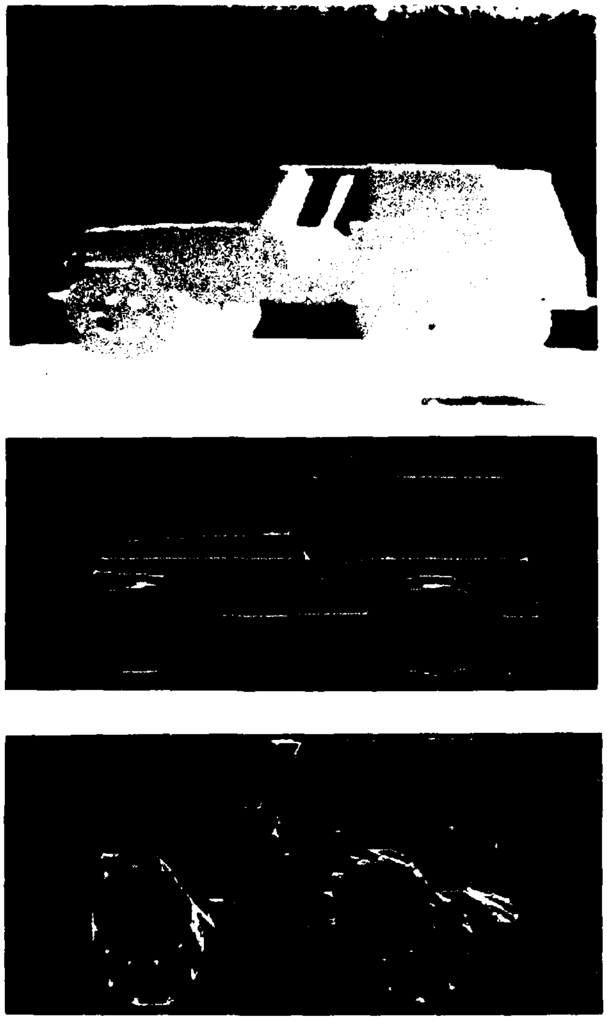


Figure 6. Range image of jeep, edges, and single view model.

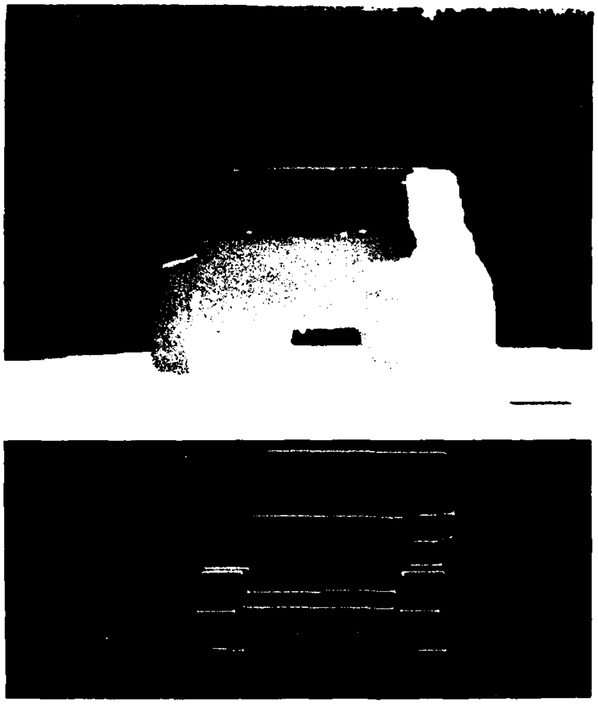


Figure 7. Second range image of jeep with edges.

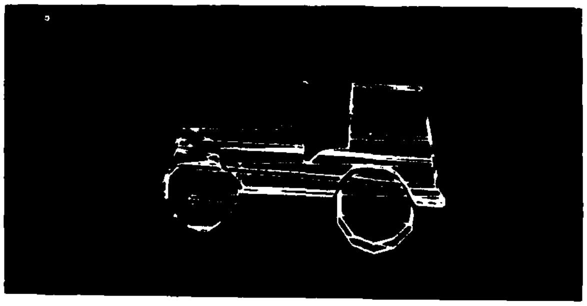


Figure 8. Multiple view model from intersection of two views.

whether for intensity or range data, results in geometrically and topologically sound volumetric models. Single view models may be integrated by repeated intersection to achieve an accurate multiple view model.

For comparison purposes, the solid models created by from intensity images and range images are compared in figure 9 (the intensity model is on the left, the range model on the right). Both models well-represent the jeep object. Similar accuracy should be available from both techniques with proper care in calibration. It is expected that the range image technique can provide extra detail in concavities not readily apparent or disambiguated by the intensity technique.

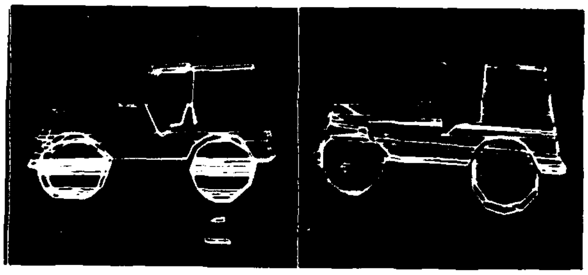


Figure 9. Comparison of composite intensity-derived model (left) with range-derived model (right).

References

1. H. Baker, "Three Dimensional Modeling," *Proc. 5th IJCAI*, pp. 649-655, Cambridge, MA, USA, 1977.

2. J. F. Canny, "Finding Edges and Lines in Images," TR 720, MIT, 1984.

3. C. H. Chien and J. K. Aggarwal, "A Volume/Surface Octree Representation," *Proc 7th International Conference on Pattern Recognition*, vol. 2, pp. 817-820, Montreal, Que., CAN, August 1984.

4. C. I. Connolly, "Cumulative Generation of Octree Models from Range Data," *Proc. 1984 IEEE Conference on Robotics and Automation*, vol. 1, p. 25, Atlanta, GA, USA, March 1984.

5. C. I. Connolly, J. L. Mundy, J. R. Stenstrom, and D. W. Thompson, "Matching from 3-D Range Models into 2-D Intensity Scenes," *Proc. 1st International Conference on Computer Vision*, London, UK, June 1987.

6. O. D. Faugeras, "New Steps Toward a Flexible 3-D Vision System for Robotics," *Proc. 7th International Conference on Pattern Recognition*, p. 796, Montreal, Que., Canada, August 1984.

7. M. Herman, "Generating Detailed Scene Descriptions from Range Images," *Proc. 1985 IEEE Conference on Robotics and Automation*, p. 426, St. Louis, MO, USA, 1985.

8. D. Lowe, "Perceptual Organization and Visual Recognition," in , Kluwer Academic Publishers, Boston, MA, USA, 1985.

9. G. Markowsky and M. A. Wesley, "Fleshing Out Wire Frames," *IBM Journal of Research and Development*, vol. 24, pp. 582-597, September 1980.

10. J. R. Stenstrom and C. I. Connolly, "Building Wire Frames from Multiple Views," *Proc. 1986 IEEE Conference on Robotics and Automation*, p. 615, San Francisco, CA, USA, April 1986.

11. J. R. Stenstrom and C. I. Connolly, "Model Generation from Images," *IV Int. Conf. Image Analysis and Processing*, Cefalu, Italy, September 1987.

12. M. A. Wesley and G. Markowsky, "Fleshing Out Projections," *IBM Journal of Research and Development*, vol. 25, pp. 934-954, November 1981.

13. J. E. Mayhew et al, "Keynote address," *IEEE Conf. on Computer Vision and Pattern Recognition*, 1986.

DEPTH FROM LOOMING STRUCTURE

Lance R. Williams and Allen R. Hanson

Department of Computer and Information Science
University of Massachusetts
Amherst, Massachusetts

ABSTRACT

A technique for computing distance to environmental surfaces suitable for obstacle recognition by a mobile robot is presented. Accurate knowledge of the camera motion parameters is not required. We demonstrate how motion in depth manifests itself in the projected lengths and areas of environmental surfaces whose extent in depth is small relative to their distance from the camera. Results on a sequence taken by a mobile robot are presented to demonstrate the accuracy of the method. Finally, we present examples of organizational principles which would allow suitable environmental structures to be recognized automatically.

1. INTRODUCTION

Perspective geometry tells us that the optical flow which confronts an organism moving through its environment is purely a function of the motion of the organism and the distance to surfaces in the environment. In principle, precise knowledge of the nature of the motion would allow its effects to be subtracted, and the distance to environmental surfaces to be computed. In this way, the organism establishes a relationship with its environment [7]. Conveniently, the motion of the organism itself, or the *egomotion*, can, in principal, be computed from the optical flow. The problem of computing the parameters of the egomotion is vastly simplified when it is known, *a priori*, that the motion is purely translational. For the case of pure translational motion, knowledge of the position of the *focus of expansion*, or *FOE*, provides two of three translation parameters, the third being velocity in the direction of gaze. There have been several attempts to use this assumption for computing distance to environmental surfaces [9,2]. Unfortunately, although this assumption is sound in theory, even a very small deviation from pure translational motion (in the form of rotation) results in significant errors in the determination of the position of the FOE (see [6] in these proceedings). This renders techniques which rely on accurate knowledge of the position of the FOE effectively useless.

Although there has been some success in computing distance to environmental surfaces assuming completely general motion [1], depth values computed in this manner are, likewise, only as accurate as the estimates of the egomotion parameters. The goal of computing distance to environmental surfaces without full and accurate knowledge of the egomotion parameters remains attractive

In this paper, we derive equations illustrating how motion in depth manifests itself in the projected lengths and areas of environmental surfaces whose extent in depth is small relative to their distance from the camera. We then present results of an experiment with an image sequence from the mobile robot domain to demonstrate the potential accuracy of the method. Finally, we present examples of organizational principles which would allow suitable environmental structures to be recognized automatically.

2. DEPTH FROM LOOMING STRUCTURE

Perspective projection can be approximated by a scaled orthographic projection when two conditions are met [11]. First, the depth to the centroid of the environmental structure in question must be large with respect to the focal length of the camera. Second, the total extent in depth of the structure must be small compared to the depth of its centroid. We call an environmental structure satisfying these two requirements a *shallow structure*. We assume that for shallow structures, scaled orthographic projection and perspective projection are equivalent. Assuming that environmental structure, of length L, satisfies the shallow structure requirement and lies at a distance, z, from the image plane, then its projected length, l_0 will be

$$l_0 = \frac{Lf}{z} \qquad (1)$$

where f is the focal length of the camera.

If the imaging device is translating into the environment with velocity, $\vec{T}$, then the component of the velocity in the direction of gaze, T_z, is

$$T_z = \vec{T} \cdot \hat{z} = |\vec{T}| \cos\theta \qquad (2)$$

where, θ is the angle between the direction of gaze and the FOE. Thus, knowledge of the position of the FOE is required only to compute the component of $\vec{T}$ in the direction of gaze. Since T_z is proportional to $\cos\theta$, and since $\cos\theta$ is essentially equal to one when the FOE and the direction of gaze are close (which is normally the case), errors of several degrees in the determination of the position of the FOE can be tolerated. After time, t, the projected length, l_1 will be

$$l_1 = \frac{Lf}{z - T_z t} \qquad (3)$$

From this, we see that the ratio, $\frac{l_0}{l_1}$ is

$$\frac{l_0}{l_1} = \frac{z - T_z t}{z} \qquad (4)$$

[1] This research is supported by the Rome Air Development Center (RADC) under contract DACA76-85-C-0008, and the Defense Advanced Research Projects Agency (DARPA) under contract F30602-87-C-0140.

Solving for z, we get

$$z = \frac{T_z t}{1 - \frac{l_0}{l_1}} \tag{5}$$

This is essentially the *time adjacency relationship*,

$$\frac{z}{T_z t} = \frac{l_1}{l_1 - l_0} \tag{6}$$

where l_0 and l_1 are not the distances from the FOE of a point at two different times, but are rather the lengths of the projection of some environmental structure. We can thus view the time adjacency relationship of [9,2] as a special case of the looming structure relationship; the FOE and the point in motion define the projection of an imaginary line segment.

We can generalize the looming structure relationship for projected lengths to projected areas. Assuming that environmental structure, with area A, satisfies the shallow structure requirement and lies at distance, z, from the image plane, then its projected area, a_0 will be

$$a_0 = \frac{Af}{z^2} \tag{7}$$

After time, t the projected area will be

$$a_1 = \frac{Af}{(z - T_z t)^2} \tag{8}$$

As with lengths, we compute the ratio, $\frac{a_0}{a_1}$ and see that it is

$$\frac{a_0}{a_1} = \frac{(z - T_z t)^2}{z^2} \tag{9}$$

Solving for z, we get

$$z = \frac{T_z t}{1 - \sqrt{\frac{a_0}{a_1}}} \tag{10}$$

3. EXPERIMENTAL RESULTS

In order to test this idea on a real image sequenoe we employed the line matching system developed by Williams and described elsewhere in these proceedings [12] to match line segments computed with Boldt's algorithm [3]. The sequence was taken with a cam ra mounted on a mobile robot and has a rotational component large enough to frustrate an algorithm dependent on accurate knowledge of the position of the FOE (Figure 1). The line matching results are satisfactory, although the lengths of the line segments produced by Boldt's algorithm (or any grouping algorithm) are often unreliable. Fortunately, the orientation and lateral placement of the lines is accurate, and we exploited this fact to define a *virtual line* whose length could be accurately measured over the course of the motion sequence. The endpoints of the virtual line are defined by the intersections of two pairs of line segments. Knowledge of the correspondence *through time of the line segments defining the virtual line*, provides information about the changing parameters of the virtual line itself. Just as a virtual line can be defined by two pairs of physical lines, *virtual region* can be defined with three or more pairs (Figure 2). For this experiment, virtual lines and regions satisfying the shallow structure requirement were defined manually, through a graphic user interface. Although the virtual lines and regions used in this experiment are either intensity discontinuities or are bounded by them, this is not a requirement. In the next section, we present a set of organizational rules which

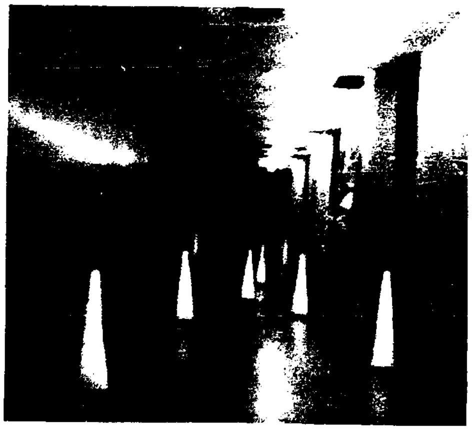


Figure 1. The first frame of a motion sequence taken by a mobile robot moving down a hallway.

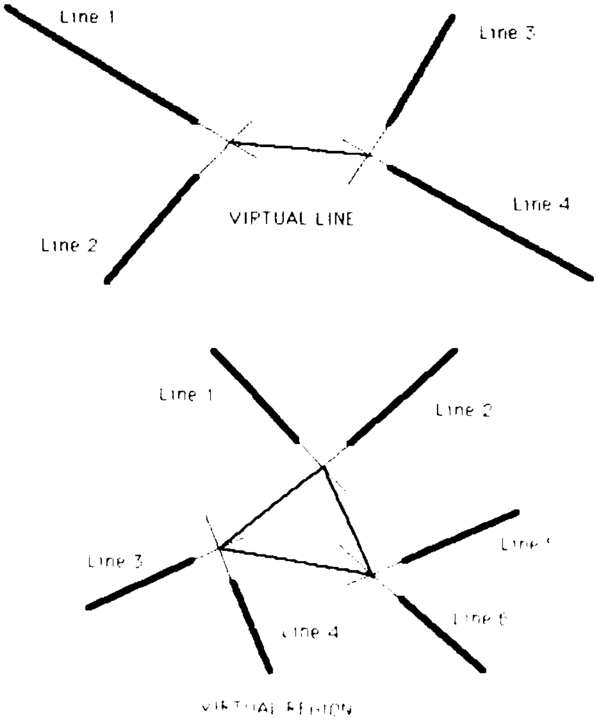


Figure 2. Defining *virtual lines* and *virtual regions* with pairs of line segments.

would allow a large number of structures satisfying the shallow structure requirement, both virtual and non-virtual, to be recognized automatically. The virtual lines and regions used, appear with labels, in Figures 3 and 4. To increase accuracy, each depth value was computed over the largest interval that all line segments defining the virtual line or region were tracked, that is to say, until one line exited the image or failed to have an acceptable match.

Knowledge of the position of the FOE improves the accuracy of depth estimates but is not critical. For this experiment, the position of the FOE was estimated by hand, although algorithms exist which are at least as accurate [9]. The robot moved a distance of 1.95 feet between frames. Knowing the position of the FOE allowed us to estimate T_z, the component of the robot's motion in the direction of gaze, as 1.91 feet. This results in less than a 3% increase in accuracy over simply assuming that the FOE and the direction of gaze are identical. The depths to the virtual lines are shown in Table 1, along with the ground truths, percent errors and the number of frames contributing to the depth estimate. Table 2 displays the same information for the virtual regions.

Table 1.

Virtual Line	Depth (ft.)	Ground Truth (ft.)	% Error	t
Cone 1	19.1	20.0	4.5	1
Cone 2	23.6	25.0	5.6	3
Cone 3	28.3	35.0	19.1	1
Cone 4	42.1	40.0	5.3	7
Can 1	29.0	30.0	3.3	7
Wall 1	27.7	27.1	2.2	2
Wall 2	48.8	48.7	0.2	7
Doorway	88.8	87.1	2.0	7

Table 2.

Virtual Region	Depth (ft.)	Ground Truth (ft.)	% Error	t
Cone 1	20.1	20.0	0.5	1
Cone 2	25.8	25.0	3.2	3
Cone 3	35.5	35.0	1.4	1
Cone 4	40.0	40.0	0.0	7

4. RECOGNIZING SHALLOW STRUCTURES

If there were no way to discriminate *environmental* structure satisfying the shallow structure requirement from structure which doesn't, then the depth from looming method would be of limited use. Fortunately, this is not the case. Organizational rules exist which would allow a perceptual organization process to automatically recognize a large class of shallow structures. All of the organizational rules which we employ are derived from two simple observations. First, certain organizations of primitive structural descriptors are unlikely to occur *through pure chance*. This is the ***non-accidentalness*** principle described by Lowe [10]. Second, since we defined shallow structures to be those structures whose perspective projections are essentially equivalent to scaled orthographic projections, we can use lack of distortion due to perspective as a means of recognizing them.

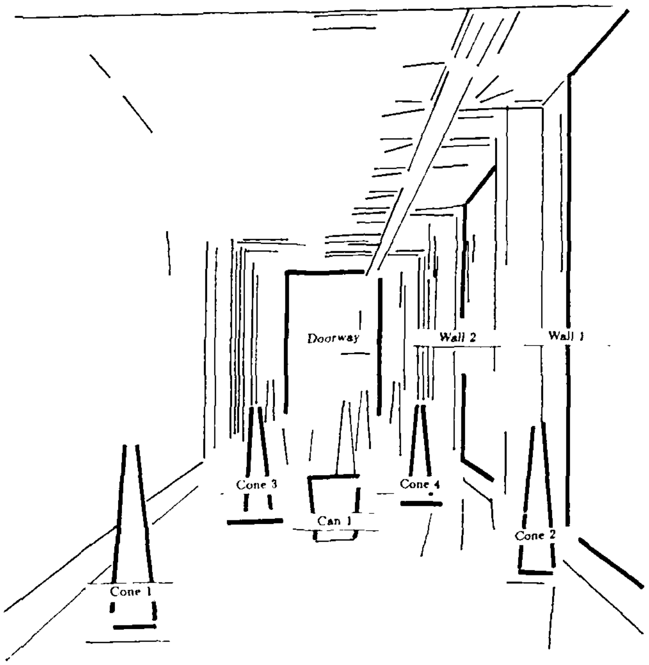


Figure 3. The line segments used to define *virtual lines*.

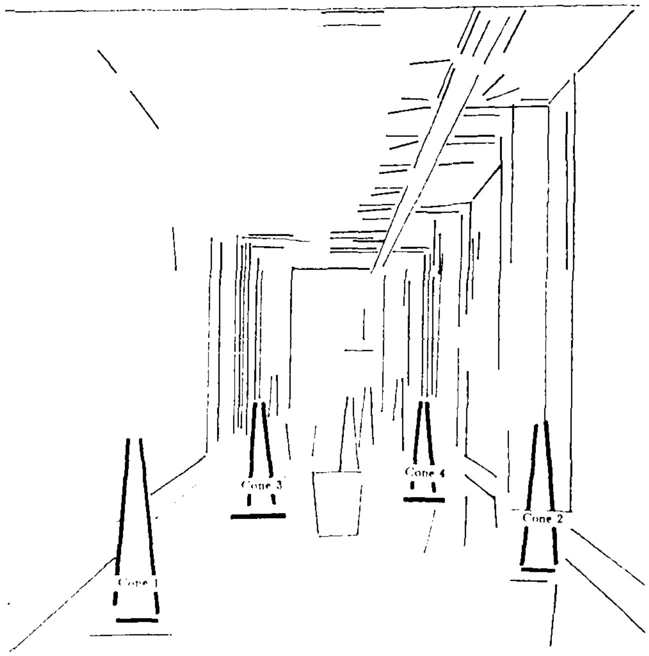


Figure 4. The line segments used to define *virtual regions*.

Spacing. Collections of equally spaced point tokens or parallel lines are highly unlikely to occur through pure chance. When we detect an arrangement of equally spaced point tokens or parallel lines in the image we can conclude not only that the corresponding environmental structures are also equally spaced [10], but also that the extent in depth of the entire collection is small compared to its distance from the camera. If this were not the case perspective effects would destroy the equal spacing (Figure 5).

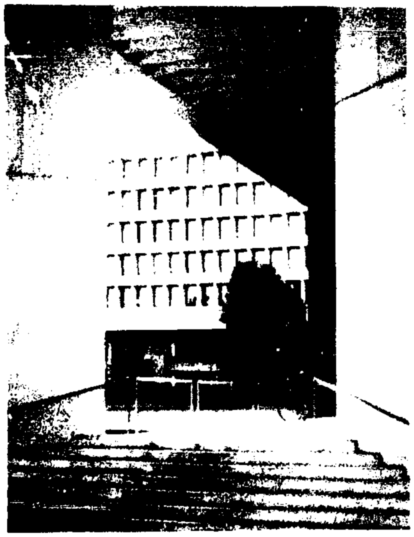


Figure 5. From the spacing of the windows on the building in the background, and of the steps in the foreground, we can infer that the distance to each structure is large relative to its extent in depth.

Parallelness. Barring accidental arrangement, parallel lines in the image are parallel in the environment [8,10] Surprisingly, we can also infer that each line individually is a shallow structure, otherwise the lines would converge due to perspective. Thompson and Mundy [11] cite this as an invariant property of scaled orthographic projections (Figure 6).

Orthogonality. Orthogonality is a compelling non-accidental relationship, especially in manmade environments. Perpendicular lines in the environment will appear perpendicular in the image only if they both lie in a plane satisfying the shallow structure requirement (Figure 7).

Symmetry. We were tempted to call this rule "non-skewed symmetry" because it represents the degenerate case of Kanade and Kender's skew symmetry method[8]. Simply stated, symmetric configurations in the image remain symmetric only if they satisfy the shallow structure requirement (Figure 8).

Some work has already been conducted at the University of Massachusetts which can be directly applied to the problem of finding shallow structures. Reynolds and Beveridge describe a method for detecting parallel and orthogonal structures in aerial images in [5]. Significantly, any line segment which is a member of some equivalence class of parallel lines or lies perpendicular to a line in such a class can be classified as a shallow structure. We also hope to employ a generic graph matcher implemented by Brolio and briefly described in [4] to recognize three line configurations exhibiting a simple bilateral symmetry.

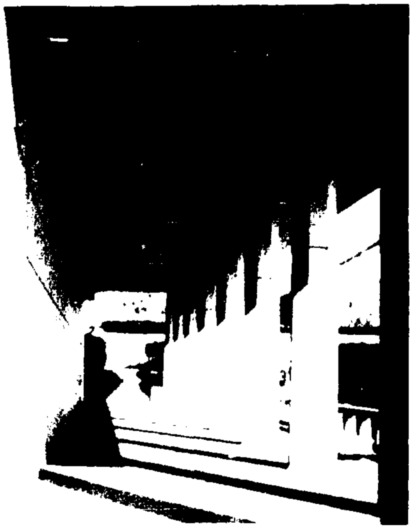


Figure 6. An equivalence class of parallel lines, each of which can be considered a shallow structure.

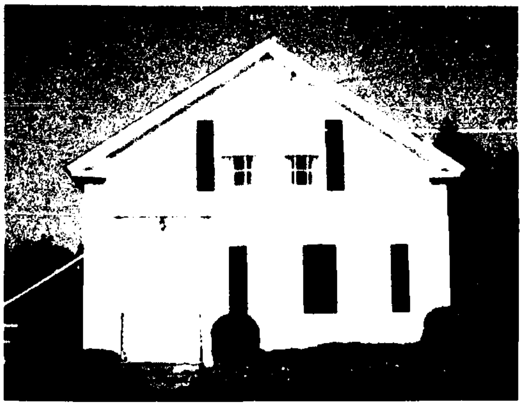


Figure 7. Perpendicular lines appear perpendicular only if they satisfy the shallow structure requirement.

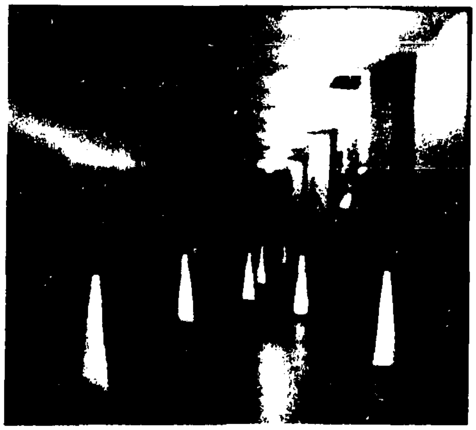


Figure 8. The traffic cones, trash can and doorway exhibit a simple bilateral symmetry that has not been distorted by the effects of perspective. We can conclude that the center line of each configuration is a shallow structure.

5. CONCLUSION

We have demonstrated that in certain cases, depth to environmental surfaces can be computed from a motion sequence without first completely determining the egomotion parameters. Depth information manifests itself not only in image plane velocities, but also in the changing lengths and areas of structural descriptors. A simple formulation for the special case of environmental structure whose extent in depth is small compared to its distance from the camera has been derived. The potential accuracy and utility of the method has been demonstrated in an experiment with an image sequence from the mobile robot domain. Although the straight line configurations employed in these experiments were defined manually, work is underway on the problem of automatically extracting these configurations from images based on general organizational principles including spacing, parallelness, orthogonality and symmetry.

ACKNOWLEDGMENTS

We wish to thank Harpreet Sawhney, Rich Weiss and Rakesh Kumar.

References

[1] Adiv, G., Interpreting Optical Flow, Ph.D. Dissertation, COINS Dept., University of Massachusetts, Amherst, Mass., September 1985.

[2] Bharwhani, S., Riseman, E. and Hanson, A., Refinement of Environmental Depth Maps Over Multiple Frames, *Proceedings of the IEEE Workshop on Motion*, 1986.

[3] Boldt, M. and Weiss, R., Token-Based Extraction of Straight Lines, COINS Technical Report 87-104, University of Massachusetts, Amherst, Mass., October 1987.

[4] Draper, B., *et al*, Tools and Experiments in the Knowledge Directed Interpretation of Road Scenes, *Proceedings of the Image Understanding Workshop*, Los Angeles, Cal., 1987.

[5] Reynolds, G. and Beveridge, J., Searching for Geometric Structure in Images of Natural Scenes, *Proceedings of the Image Understanding Workshop*, Los Angeles, Cal., 1987.

[6] Dutta, R., *et al*, Issues in Extracting Depth from Approximate Translational Motion, *Proceedings of DARPA Image Understanding Workshop*, Cambridge, Mass., 1988.

[7] Gibson, J., *The Senses Considered as Perceptual Systems*, Houghton-Mifflin, Boston, Mass., 1966.

[8] Kanade, T. and Kender, J., Skewed Symmetry: Mapping Image Regularities Into Shape. CMU-CS-80-133, Computer Science, Dept., Carnegie Mellon University

[9] Lawton, D., Processing Dynamic Image Sequences from a Moving Sensor, Ph.D. Dissertation, COINS Dept., University of Massachusetts, Amherst, Mass., February 1984

[10] Lowe, D.,*Perceptual Organization and Visual Recognition*, Kluwer Academic Publishers, Boston, Mass., 1985

[11] Thompson, D. and Mundy, J., Three Dimensional Model Matching From an Unconstrained Viewpoint, *Proceedings of the IEEE Conference on Robotics and Automation*, Raleigh, N.C., 1987.

[12] Williams, L.R. and Hanson, A.R., Translating Optical Flow into Token Matches, *Proceedings of DARPA Image Understanding Workshop*, Cambridge, Mass., 1988.

On the Recovery of Superellipsoids

Terrance E. Boult and Ari D. Gross
Columbia University Department of Computer Science
New York City, New York, 10027. tboult@cs.columbia.edu, ari@sylvester.columbia.edu

Abstract

Superellipsoids are parameterized solids which can appear like cubes or spheres or octahedrons or 8-pointed stars or anything in between. They can also be stretched, bent, tapered and combined with boolean to model a wide range of objects. Columbia's vision group is interested in using superquadrics as model primitives for computer vision applications because they are flexible enough to allow modeling of many objects, yet they can be described by a few (5-14) numbers. This paper discusses research into the recovery of superellipsoids from 3-D information, in particular range data.

This research can be divided into two parts, a study of potential error-of-fit measures for recovering superquadrics, and implementation and experimentation with a system which attempts to recover superellipsoids by minimizing an error-of-fit measure.

This paper presents an overview of work in both areas. Included are data from an initial comparison of 4 error-of-fit measures in terms of the inter-relationship between each measure and the parameters defining the superellipsoid. Also discussed is an experimental system which employs a nonlinear least square minimization technique to recover the parameters. This paper discuss both the advantages of this technique, and some of its major drawbacks. Examples are presented, using both synthetic and range-data, where the system successfully recovers superlliposids, including "negative" volumes as would occur if superellipsoids were used in a constructive solid modeling system.

1 Introduction

As a modeling primitive, superquadrics provide a natural extension to traditional CAD models, [Barr 81]. Recently they have become the focus of a few computer vision research efforts, see [Pentland 86a], [Pentland 87], [Bajcsy and Solina 87b], [Solina 87], and [Boult and Gross 87b]. Most of the research to date has concentrated on a subclass of superquadrics called superellipsoids (hereafter SEs). The growing research interests in SEs can partially be attributed to the fact that they can mocel a wide range of object in a very compact form. Another contributing factor is the underlying mathematical formulation which provides convenient tools for their recovery.

The next section of this paper presents a mathematical definition of superellipsoids and a little motivation for using SE's in computer vision. Section 3 defines various error-of-fit measures which could be used in the recovery of SE's, and presents an initial comparison of these measures.

The current system is described in Section 4. Briefly it employs a nonlinear least square minimization technique on the inside-outside function to recover the parameters. Oddly, this function is one of the poorer error-of-fit measures considered, yet it is sufficient to allow for reasonable (qualitive) recovery of SEs from both synthetic and range data. A few examples of the results of the system are then presented. The examples include the use of very sparse data, synthetic multiple views and the recovery of a negative superellipsoid from range data.

Previous papers by the authors, [Boult and Gross 87b] and [Boult and Gross 87a], discuss a few other difficulties that are encountered when modeling objects with superellipsoids, or attempting to recover superquadrics from 3D data. These difficulties include the general problems associated with non-orthogonal representations, the difficulties of dealing with objects which are not exactly representable with CSG operations on the primitives, the need to recognize negative objects, and certain numerical instabilities.

Some pros and cons of the approach as well as few conclusions, and a discussion of work in progress at Columbia's vision lab appear in the final section.

The main result in this paper is that there are problems and biases with currently used error-of-fit measures, as well as with two newly proposed measures. It also suggests that even better measures await definition, especially ones measuring error using knowledge of the sensor direction. Another important result of the research is that least square minimization, (even using a poor error of fit measure), allows recovery of both positive and negative instances of superellipsoids from depth data.

2 Background and Motivation

Mathematically, SE solids are a spherical product of two superellipse curves (see [Gardiner 65]). Superellipse curves are similar to traditional ellipses except the terms in the definition are raised to parameterized exponents (not necessarily integers), i.e.: $(\frac{x}{a})^{\frac{2}{\epsilon}} + (\frac{y}{b})^{\frac{2}{\epsilon}} = 1$. When ϵ, the *relative shape parameter* is 1, the curve describes an ellipse. As the relative shape parameter varies from 1 down to 0, the shape becomes progressively squarish; as it varies from 1 toward 2, the shape transforms from a ellipse to a diamond shaped bevel. When the parameter is greater than 2, the shape becomes pinched and as the parameter approaches

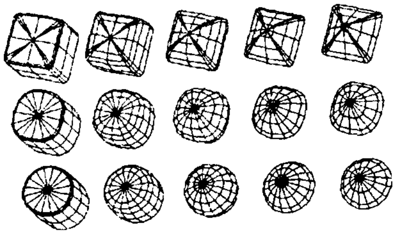


Figure 1: Superellipsoids with relative shape parameter ϵ_1 having values .1, .5, 1, 1.5 and 2 (left to right), and ϵ_2 having values .1, .6 and 1 (top to bottom).

infinity, the shape approaches a cross.

The result of the spherical product of two such curves is conveniently represented in a parametric form, e.g., a superellipsoid can be represented as:

$$\vec{s}(\eta,\omega) = \begin{bmatrix} a_x \cdot \cos^{\epsilon_1}\eta \cdot \cos^{\epsilon_2}\omega \\ a_y \cdot \cos^{\epsilon_1}\eta \cdot \sin^{\epsilon_2}\omega \\ a_z \cdot \sin^{\epsilon_1}\eta \end{bmatrix}, \quad \begin{matrix} -\frac{\pi}{2} \leq \eta \leq \frac{\pi}{2} \\ -\pi \leq \omega \leq \pi, \end{matrix} \qquad (1)$$

for any fixed positive$a_x, a_y, a_z, \epsilon_1$, and ϵ_2.

The parameters a_x, a_y, a_z affect the size of the superellipsoid along the x, y and z axes (respectively) in the object centered coordinate system. The parameters ϵ_1 and ϵ_2 effect the relative shape of the superellipsoid in the latitudinal (xz) and longitudinal (xy) directions, see figure 1. When the 5 parameters are all unity, the superellipsoid is the unit sphere.

Superellipsoids can also be defined implicitly as the the volume where the SE inside-outside function is negative, where the SE inside-outside function is given by:

$$f(x,y,z) = \left(\left(\frac{|x|}{a_x}\right)^{\frac{2}{\epsilon_2}} + \left(\frac{|y|}{a_y}\right)^{\frac{2}{\epsilon_2}}\right)^{\frac{\epsilon_2}{\epsilon_1}} + \left(\frac{|z|}{a_z}\right)^{\frac{2}{\epsilon_1}}$$

$$\text{where} \qquad (2)$$

$$\text{if} \begin{cases} f(\hat{x},\hat{y},\hat{z}) = 1, \text{ then } \hat{x},\hat{y},\hat{z} \text{ is on the surface,} \\ f(\hat{x},\hat{y},\hat{z}) < 1, \text{ then } \hat{x},\hat{y},\hat{z} \text{ is inside the volume,} \\ f(\hat{x},\hat{y},\hat{z}) > 1, \text{ then } \hat{x},\hat{y},\hat{z} \text{ is outside the volume.} \end{cases}$$

These equations define superellipsoids (hereafter referred to as SEs) which are probably the simplest of a family of surfaces called superquadrics. SE's are the only superquadrics which can be made into a convex solid. One can also define superhyperboloids of one or two sheets and supertoroids, see [Barr 81]. In addition to the variety of shapes defined by the basic superquadrics, Barr also discusses the application of angle-preserving transforms which allow translation, rotation, bending and twisting. With the addition of tapering and traditional boolean combination operations, superquadrics become a powerful modeling tool, see Figure 2 for some exemples generated using SuperSketch, [Pentland 86b].

While translation, rotation, bending and tapering are important modifications to basic SE primitives, they will not be considered in the remainder of the paper. Future work will examine the effect of these deformations on error-of-fit measures and potential implementations.

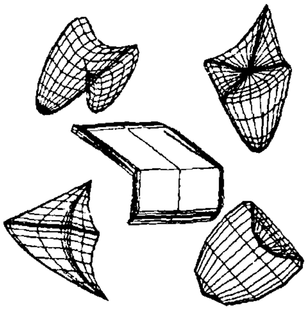


Figure 2: Examples of SE's deformed by bending and tapering

Definitions in hand, let us now discusses some of the rational for using SE's for object recognition. In short, they are flexible enough to represent a wide class of objects, but are simple enough to be recovered from 3-D data. Their functional form and inside-outside function provides a useful tools to aid in their recovery.

Traditional constructive solid modeling systems use boolean operations to combine primitives, such as, spheres, cylinders, and rectangular solids. By extending the primitive shapes to SEs, we allow the user to easily produce a continuum of forms, from spheres, to cuboids with rounded corners, to cubes, to diamond shapes. Such objects are difficult to model with traditional constructive solid geometry (hereafter CSG) systems. Making them primitives simplifies the job of the designer for any problem that contains objects with these properties.[1]

However, adding flexibility above that of a traditional CSG system is not the only reason to choose superquadrics. For added flexibility, one could make the primitives of the system Generalized Cylinders or Generalized Cones, see [Brooks and Binford 80]. The problem with GCs is that recovering them is a difficult process, partially because each GC may require hundreds of parameters to describe it. The number of parameters necessary to define a GC depends on the complexity of the cross section function, the spine, and sweeping function. As in [Brooks 85], [Rao and Nevatia 86], or [Ponce, Chelberg and Mann 87], one generally has to greatly restrict the class of GCs allowed before one can reliably recover them.

It is interesting to note that superellipsoids and supertoroids can be defined as a subclass of GC; SEs are those straight spined GCs with their cross section and sweeping functions defined by superellipses.[2] If the SEs are bent or tapered, these defoimations are applied to the spine and sweeping function respectively.

[1] A simple CAD system has been developed by A. Pentland, see [Pentland 86b] that combines SEs (constrained to only those that are convex solids) with CSG type operations plus bending and tapering. The ease with which people became accustomed to modeling with this system may be tied to their own internal representation of the objects. [Pentland 86a] presents arguments to motivate the use of superquadrics as modeling primitives by showing correspondences to human vocalization of object descriptions.

[2] The sweeping rule is highly nonlinear, and always goes to zero. The authors do not know of any system which can recover a subset of GC containing this class.

Thus superquadrics provide a restriction which may allow efficient recovery as well as a simpler set of "limb equations"(after [Ponce, Chelberg and Mann 87]). In addition, they provide mathematical properties (the explicit parametric definitions, inside-outside function (implicit definition) and a normal surface duality principle) which do not exist for most classes of GCs.

The inside-outside function provides a useful tool for recovery, because it provides a simple way to determine which of the data points are inside the surface. Furthermore, the value of the function grows as points are moved further from the surface. Thus it can be used in conjunction with a minimization technique to recover the surface. However, the inside-outside function is not a euclidean "error of fit" measure (see the following section), and therefore direct minimization of it will not necessarily provide satisfactory results. (However, in two independent implementations this technique has produced satisfactory results, see [Bajcsy and Solina 87b] and [Boult and Gross 87b], and section 4.)

Canonical superquadrics (except the supertoroids) have a desirable duality property. The superquadric normal vectors lie on a dual superquadric form; that is, if the normal vectors were translated to the origin, they would generate another superquadric of the same class such that the normal of the new surface (if translated to the origin) would produce a copy of the original superquadric (except for a translation). One can easily derive the form of the normal surface as a spherical product as well as its own inside-outside function, see [Barr 81]. Note that the *inside-outside function for the surface normals* might be used to find superquadrics fitting surface normal information as would be available from shape-from-X methods.

3 Comparison of four "Error of fit" measures

The recovery of superellipsoids from depth data has recently been the subject of research in various labs, e.g. see [Pentland 87], [Bajcsy and Solina 87b], and [Boult and Gross 87b]. Each of the approaches has something in common: at some point they define some measure of the "error of fit" (hereafter EOF), and they use a nonlinear minimization technique to recover the parameters of the superellipsoid starting from an initial estimate.[3] It is well known, from work on fitting curves to points in the plane, that different EOF functions can result in radically different curve estimations, e.g. see [Pratt 87], [Sampson 82], [Bookstein 79] and [Turner 74]. On close examination of Columbia's initial system to recover SEs, it became apparent that the EOF function being used was not even proportional to sensor errors (determined using synethic data). The purpose of this section is to examine some of the measures of the error of fit used by these researchers as well as some measures not yet used in minimization procedures. While rotation and translation are not going to be considered herein, they are easily added to any of the measures under consideration.

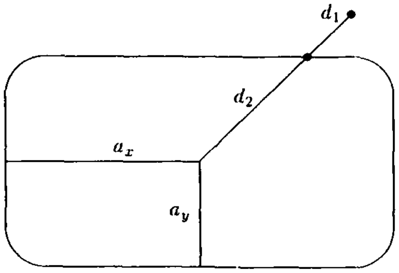


Figure 3: Drawing showing the intuitive meaning of measures. (This is an x-y cross section of an SE) $EOF_1 \approx \left(\frac{d_1}{d_1+d_2}\right)^{\frac{2}{\epsilon_1}}$, $EOF_2 \approx \left(\frac{d_1}{d_1+d_2}\right)^2$, $EOF_3 \approx (a_x \cdot a_y \cdot a_z) \cdot \left(\frac{d_1}{d_1+d_2}\right)^2$, $EOF_4 = d_1$

In particular, this section considers 4 measures of error of fit. This section defines each of these mathematically and also gives a brief intuitive description in 2 dimensional physical/geometrical terms.

3.1 EOF_1: Mean-square-value of the inside-outside function

The first measure to be considered is, in some sense, the simplest. It is directly based on the inside-outside function which can be used to implicitly define the SE.

$$EOF_1 = \frac{1}{n} \sum_{i=1\ldots n} \left(1 - \hat{f}(x_i, y_i, z_i; \epsilon_1, \epsilon_2, a_x, a_y, a_z)\right)^2 \qquad (3)$$

where x_i, y_i, z_i are the given depth data points converted to a single viewpoint, and where $\hat{f}(x_i, y_i, z_i; \epsilon_1, \epsilon_2, a_x, a_y, a_z) = f(x, y, z)$ where $\epsilon_1, \epsilon_2, a_x, a_y, a_z$ are parameters in equation 2. A rough intuitive definition of this measure, see figure 3, is as a power of the ratio of the length of two line segments. The exponent to which the ratio is raised depends on the squareness-roundness in the xz direction and is given by $\frac{2}{\epsilon 1}$. The denominator of the ratio is the length of the line segment between the data point and the center of the SE. The numerator is the length of that portion of the above mentioned line segment connecting the center of the SE with that point on the surface in the direction of the data point. Given this intuitive definition, it is obvious that the measure can decrease if the object size grows even if the actual distance to the surface increases. Additionally, for small values of ϵ_1, this measure can be quite large (or small depending on ratio), and computationally unstable. This will be borne out through the computational evaluation in later sections.

[3] There are many differences in the details of the algorithms. The work of [Bajcsy and Solina 87b] and [Boult and Gross 87b] differ mostly in minor de- t .ils, and both are driven mainly by the minimization approach with only minor effort to derive good initial estimates of the parameters. The approach in [Pentland 87] is radically different using a detailed nonadaptive initial search of the parameter space (at 64,000 locations), but still maintains the minimization of some error of fit measure after that initial search through the parameter space

3.2 EOF_2: "Corrected" mean-square inside-outside measure

The second measure to be considered is a modified version of the first to remove some its most undesirable properties: the rapid grown of EOF_1 for small values of ϵ_1, and some of its biases. The new measures is given by:

$$EOF_2 = \frac{1}{n} \sum_{i=1 \dots n} (1 - \hat{f}^{\epsilon_1}(x_i, y_i, z_i, \epsilon_1, \epsilon_2, a_x, a_y, a_z))^2 \qquad (4)$$

where terms are as defined for EOF_1

The intuitive definition of this measure is as the same ratio as for EOF_1 except that the power to which the ratio is raised is simply 2. Thus, this measure gives a type of "relative error" in a radial direction, however, it greatly effects measurements in the z direction (the drawing above ignores that direction). As before, it is obvious that the measure can decrease if the object size grows even if the actual distance to the surface increases. Note that to date this particular measure has not been used in a SE recovery system.

3.3 EOF_3: a " minimal volume" function

Another EOF function, used by [Bajcsy and Solina 87b], is

$$EOF_3 = \frac{1}{n} \sum_{i=1 \dots n} \left(\sqrt{a_x \cdot a_y \cdot a_z} \cdot (1 - \hat{f}^{\epsilon_1}(x_i, y_i, z_i, \epsilon_1, \epsilon_2, a_x, a_y, a_z)) \right)^2 \qquad (5)$$

where terms are as defined for EOF_1.[4]

A variant of this error of fit measure was heuristically first motivated in [Bajcsy and Solina 87b] by the idea that there are many surfaces which might fit data from a single view, and the system should recover the surface of minimal volume. The actual equations 10 and 11 in [Bajcsy and Solina 87b], as well as similar equations in [Bajcsy and Solina 87a], suggest a multiplicative factor of $a_x \cdot a_y \cdot a_z$ rather than $\sqrt{a_x \cdot a_y z \cdot a_z}$. However, this is not in keeping with the heurstic definition, and the latter form appears in [Solina 87]. While not reported here, experimental comparisons have found the the factor of $a_x \cdot a_y \cdot a_z$ results in a meausre wchich has similar properties, but exaggerates the poor behavior of EOF_3.

The rough intuitive definition of EOF_3 is simply the square of the "relative error" in the radial direction multiplied by the square-root of volume of the cube bounding the SE (which is meant to be a crude approximation to the SE itself). The advantage of this is that in a pure relative error measure, the absolute error in the radial direction was divided by a term containing the the radial distance to the surface in some direction. Thus with the points inside the object, the object could grow and reduce EOF_1 or EOF_2 (with respect to one point) while the absolute error would grow. By multiplying by the volume of the bounding rectangular solid, this behavior is curtailed. However, this behavior is replaced with the possibility of reducing the measure by keeping the relative error in the radial direction constant, but reducing the size of the SE (i.e., the measure is sensitive to scale).

3.4 EOF_4: A measure based on true euclidean distance

A final test measure is defined as the mean distance between each data point and the corresponding point on the surface of the superquadric on the line connecting the data point and the center of the SE. In equation form this becomes:

$$\begin{aligned} EOF_4 &= \tfrac{1}{n} \textstyle\sum_{i=1 \dots n} |\vec{x_i} - \vec{q}| \\ &= \tfrac{1}{n} \textstyle\sum_{i=1 \dots n} \sqrt{(x_i - q_{i_x})^2 + (y_i - q_{i_y})^2 + (z_i - q_{i_z})^2} \end{aligned} \qquad (6)$$

where

$$\begin{aligned} \vec{q_i} = \vec{s}(\eta, \omega),\ \omega &= \left(tan^{-1} \left(\tfrac{a_x \cdot y_i}{a_y \cdot x_i} \right)^{\frac{1}{\epsilon_2}} \right), \\ \eta &= \left(tan^{-1} \left(\tfrac{a_x \cdot z_i}{a_z \cdot x_i} \cdot sin^{\epsilon_2}(\omega) \right)^{\frac{1}{\epsilon_1}} \right), \end{aligned} \qquad (7)$$

and $\vec{s}(\cdot, \cdot)$ is defined as in equation 1, and the relative signs of x_i, y_i, z_i, the i^{th} data point, are used to determine the correct spherical quadrant of the arctan.

This measure overestimates the minimal distance of a point from the SE, especially for squarish ones. However, it is a true euclidean distance measure, and thus is not effected by overall scaling of the problem. There are probably better euclidean based measures, however, the only better measures currently known to the authors do *not* have a functional form defining them, and thus are difficult to minimize.

3.5 Evaluation of the Different Error of Fit Measures

This section begins the analysis of the various error of fit measures. The main vehicle for presentation will be graphical, with running commentary giving *one* interpretation of the results. The reader is advised to draw her/his own conclusions.

The evaluation is in terms of two properties of the measures: parameter bias, and shape of crossections of the error surface. The parameter bias makes it difficult to compare two potential solutions and also generally affects the minimization by shifting the location of the minima. The relation of the measure (in terms of the scale of values) to the RMS error of data is a by product of the experiment to determine bias. The shape of cross sections of the error surface is important because it affects both the possibility of minimization (in particular, is it relatively concave?), and the rate of convergence of iterative minimization techniques (how large is the gradient?).

3.5.1 Determination of Measure Bias

To compare the parameter bias, each measure will be computed using the "correct" values of the 5 parameters (i.e. the parameters used to generate a surface). Of course, because of the noise in the

[4] Actually, there a few minor differences between this definition and the ones in [Bajcsy and Solina 87b]. The first difference is the multiplication by $\frac{1}{n}$ which normalizes by the number of data points. The second difference is that they use a homogeneous transformation to deal with rotation and translation, thus the derivation of EOF_1 in that paper is not given in terms of the $\hat{f}$ above, but this formulation can be easily shown to be equivalent.

data points, these parameters will generally not be those parameters for which a given measure is minimized. (More about that later). The different cases considered in this section should give the reader a good intuitive feeling for how the different measures are biased (with respect to the true RMS error) as the parameters defining an SE are varied. To cut down on the amount of analysis, certain symmetries have been anticipated, in particular a_y is assumed to be identical to a_x. The discussions in this section are a review of the computational experiment, and only represent a small sample of the experimental data.

Each of the "bias" graphs show a sequence of computational experiments where a single parameter is varied over some range. The values are scaled, and the pattern of the scaled values often provide a good feel for the bias of measure. The scale values themselves show the relative difference in the size of the error measure (using the "correct" answer) with respect to the RMS error added to the data.

The data in each graph was computed as follows:

Step 1 An initial set of parameters for the SE, error range, and sensor direction (x, y, or z) are chosen. (Only x direction presented here). Also, a parameter is chosen to vary in the iterations. Each iteration generates one data point on the graph for each of the different measures, computed in steps 2 through 5.

Step 2 For a given set of SE parameters (i.e. each iteration), 500 points are generating on the surface of the SE assuming a uniform pseudo-random distribution of angles on the viewing hemisphere for the chosen sensor direction.

Step 3 For each data point, a pseudo-random noise value is computed assuming a uniform distribution in a predetermined range. The noise is added to the data point in the direction of the sensor. The value is directly added to the variable used to compute the RMS measure.

Step 4 Using the exact value of the set of parameters for this iteration, and the noisy data computed in steps 2 and 3, the value of each of the measures EOF_1, EOF_2, EOF_3 and EOF_4 is computed and recorded.

Step 5 For each measure (including RMS) the recorded values are examined and the maximum of a measure is used to re-scale its value into [0,1] and the value is entered on the graph.

Before beginning the description of the initial results, it is necessary to make a few comments about the graphical presentation of the results. First note that each graph contains 5 different marks which are used to show the scaled value of the measures. Also, each graph has a legend *with scaling factors*. Thus in figure 4, the computed value of EOF_1 for $\epsilon_{xy} = .5$ is approximately .64 x .176, and when $\epsilon_{xy} = 1.8$ is approximately .91 x .176. Looking at the tick-marks allows one to obtain a feel for the measure's bias as the parameter varies, and the scaling provides a means of relating the measure to the actual RMS error. For space considerations, only a few examples are presented here. However, the running commentary in this section is based on the analysis of over 500 of the above computational experiments.

The interest in bias factors stems from their importance in comparing the realtive quality of two potential solutions. For example, if the bias in a parameter P is toward larger values then if two candidate sets of parameters differ only in parameter P, the it can occur that the set with the larger P value may report a larger measure but have a smaller actual RMS error. Thus to properly compare two potential solutions, one needs to be able to estimate the bias in each of the parameters in which they differ.

The reader should recall that these graphs are *not* showing the shape of the error surface near the correct solution, and the non-monotonic behavior does not mean that minimization will be difficult.

The first graphical display for the bias experiment is in figure 4, and involves the bias of the measures with respect to the parameter ϵ_{xy}, the squareness-roundness in the xy direction. The first thing that should strike you about the graph is that all of the measures clump together, and none of them show a particularly smooth bias. There is a general trend toward having a large measure for ϵ_{xy}, but locally it is far from monotonic. Thus, the result of the minimization process on all the measures will generally report some value which overestimates the error (even if it could be scaled) and the overestimate is a generally increasing (but nonmonotonic) function ϵ_{xy}.

One interpretation of the clumping of the measures is that the observed bias is caused by measurement in the radial direction when the sensor error is in an axial direction. This interpretation suggests that no radial measure will be bias-free and provides an impetus to attempt to derive measures in the sensor direction.

The scales of the measured values were generally close to the RMS error, with EOF_1 very close (by accident), and EOF_3 EOF_4 off by a factor of 2. Although not evident from the examples presented here, EOF_3 is greatly effected by scaling. If all sizes in the example were multiplied by a factor of 100 (say meters into mm) then EOF_3 would increase by a factor of 1000000. On the otherhand, EOF_1 would decrease (by a much smaller amount) as the scale was increased.

The second example in the bias determination experiment is shown in figure 5. The most striking feature of this graph radically different behavior of the measures. Note that EOF_4 follows the RMS error rather well, showing that it is not biased with respect to ϵ_{xz}. EOF_2 and EOF_3 (which differ only in scale) show the same nonmonotonic but generalling increasing bias as they did for ϵ_{xy}. (Close examination shows qualitatively similar small scale perturbations in EOF_2, EOF_3 and EOF_4. Again, these are probably biases inherent in any axial measure.) While the other measures show a bias for larger values, EOF_1 has a significant, and apparently monotonic, bias for smaller values.

A second fact to consider in this example is the relative scales of the measure. A glance at the scaling factors in the legend suggest that only EOF_4 is closly linked to the actual RMS error. The other measures are all off by at least an order of magnititude, or more.

The third example in the bias determination experiment is shown in figure 6. The most striking feature of this graph is the clear separation of the measures. EOF_2 tracks the RMS error rather well, showing that it is not biased over this range of a_x.

EOF_4 is also rather unbiased, with the exception of very small values of a_x, which have a negative bias. Both of the aforementioned measures show non-monotonicity in what bias they do have, but the deviations (when appropriately scaled) from the RMS error are not very significant. EOF_1 on the other hand, shows a strong, almost monotonic, bias toward smaller values of a_x, as does EOF_3. The bias of EOF_3 for smaller values might be attributied to the "volume" factor which is the only difference between it and EOF_2. While not shown here, the above patterns are still present when the sensor direction is changed to the z axis, although the rates of change of the bias (i.e. slopes of curves) are smaller.

The final example from the bias experiment, is shown in figure 7. Again there is a rather noticeable difference between the measures. This time it is EOF_1 which display virtually no particular bias, although there are some obvious local non-monotonicities. The other three measures all show a strong bias for larger a_z values, with the bias in EOF_3 slightly less than in the other measures. While one might have predicted a bias in EOF_3 for small values of a_z, the greater bias of the underlying EOF_2 factor dominates the "volume" term (but the latter probably provides the mitigating factor in the bias as compared with EOF_2 and EOF_4). To the eye, the bias trend in all but EOF_1 appear linear, and anticipation/compensation might be possible. Again, the experiments with the sensor direction along the z axis have similar overall characteristics, though the slopes of the bias "curves" were smaller.

As mentioned, there were also numerous test cases included in the experiment which were not reported here. To summarize the results, little characteristic difference was found to be caused by changes in the sensor direction, although many minor variations did occur. As mentioned above, many of the measures also displayed a bias in terms of overall scale of the problem.

A final comment about the bias experiments. The graphical presentation makes it very easy to overlook the absolute scaling parameters in the legend. However, if one were interested in converting from one measure into an approximate RMS measure, even knowing the bias characteristic would not be useful unless one could also compute the scaling factors. As shown in the next section, many of the measures can have significant scale changes in the neighboorhood of the correct solution, thus estimation of the scale factors from this experiment is unlikely.

3.5.2 Analysis of cross sections of error-of-fit surface

This section discusses experiments which attempts to determine the structure of the error-of-fit surface by examining one dimensional slices through that multi-dimensional surface. The experiment considered both the structure in a small neighborhood of the correct parameter value and the larger scale structure (say variations of 10-100%). The local structure provides insight into how the measures err (i.e. it shows location of measures minima). The large scale structure provide insight into the likelyhood of successful minimization when reasonable initial estimates are used as starting values. Because global properties are of more importance, only the experiments using a wide range of values will be reported here, even though some of the conclusion are more readily made by examination of the experiments in small neighborhoods.

As in the reporting of the bias experiments, the presentation will be graphical with commentary. Each graph was generated as follows:

Step 1 Parameters defining SE and sensor direction are chosen. Using these parameters, 500 points are generated on the surface of the SE assuming a uniform pseudo-random distribution of angles on the viewing hemisphere for the chosen sensor direction. For each data point, a pseudo-random noise value is computed assuming a uniform distribution in a predetermined range. The noise is added to the data point in the direction of the sensor. The value is directly added to the variable used to compute the RMS measure.

Step 2 A parameter is chosen to vary, as well as the range of variation. For each iteration point (10-20 spanning the chosen range) the experimental values for each measure are generated using the current value of the variable parameter, and the correct values of the remaining parameters, and are recorded.

Step 3 For each measure (including RMS) the recorded values are examined and the maximum of a measure is used to rescale its value into the interval [0,1] and the value is entered on the graph.

Note the above procedures result in the correct location of the minima occuring in the center of each graph.

Probably the most surprising result is that when rotations and translations are ignored (as in the results reported here), all of the measures generated relatively concave cross sections. Thus, minimization of any of them (given correct rotations and translations) should be straight forward. The main difference between the measures was the degree of concavity, which would effect the efficiency of the minimization.

The first presented crossection, figure 8, shows the cross section as ϵ_{xz} is varied. The clear separation of EOF_4, and higher curvature suggest that it is the preferred measure in this example. Note that since a_x, a_y and a_z are fixed, the structure of EOF_2 and EOF_3 is identical; they differ only in a scaling factor. Finally, EOF_1 does show some concavity, but the dramatic scale differences make the minima very shallow. While not shown, the separation of the measures is even greater when the errors are along the z axis.

It is worth pointing out that EOF_2, EOF_3 and EOF_4 all have their minima to the left (i.e. smaller parameter value) than the underlying surface, while EOF_1 overestimates the parameter. (The detailed analysis of the local region which is not presented shows that the minima of EOF_4 is the closest to the parameter value defining the actual underlying surface.) The reader might want to compare this with the results of the bias experiment for the same parameter.

The second cross section presented in figure 9, shows all of the measures are very asymetric about their minima. In addition, although it cannot be seen from this graph, all of the measures have their minima above the "correct" value. The dramatic changes in scale for EOF_1 make it appear flat on this graph, but it does have a shape similar to the other measures, just more dramatic.

It is also obvious (though barely visible) that EOF_4 has the sharpest minima, although minimization of any of the measures with a starting value above the true solution looks like a good idea. Again, EOF_4 has its minima closest to the correct value. The asymmetric shape of this curve is even more pronounced when the errors in the data are along the z axis.

A comment also is in order regarding the vast differences in scales between the measures. As one can see from the legend in figure 9, the relative scales differ by orders of magnitude. From a practical point of view, those measures which routinely have a wide numerical range (EOF_1, EOF_2 and depending on scaling possibly EOF_3) present more numerical problems, and show a greater sensitivity to roundoff errors.

A third cross section, see figure 9, shows a case where the relative shape of all of the measures is almost identical, with EOF_4 just slightly outperforming the other measures in the concavity analysis. Also, note that all of the measures again overestimate the correct underlying parameters (by about 1%-2%). When the error was in the direction of the z axis, all measures were very sharply concave, and the minima were very close to the correct value.

The final cross sections, see figures 11 and 11 show the structure of the error surfaces for some of the parameters used in the bias experiments. Note that in these cases, EOF_4 is a clear winner.

4 The Current System

This section presents some of the details of our current system for the recovery of superquadrics from 3-D information. The basis of our system is the minimization of EOF_1. Note that one of the advantages of minimizing the a measure based on the inside-outside function is that it requires little extra effort to incorporate multiple views, assuming one knows the sensor position for each view (to convert points to a common coordinate system). However, such an approach will not take error distributions into account, and the errors in conversion may be exaggerated by the fitting process.

Given 3-D information about a SE in canonical position, one can use a nonlinear minimization technique to recover the 5 parameters needed to define it. Our system uses a Gauss-Newton iterative nonlinear least square minimization technique, (for example see [Hageman and Young 70]). If the SE is not in canonical position, the system must also recover estimates of the translation and rotation necessary to put the information on the surface of a canonical SE. There are obviously many approaches to deal with the translation and rotation. The two most obvious are: the use of a pair of transforms (one for translation and one for rotation) and the use of a homogeneous transform that combines both translation and rotation. Our system uses the first approach.

For the remainder of this section let $C_\theta = \cos\theta$ and $S_\theta = \sin\theta$. Thus given a canonical SE surface defined as $\vec{s} = \vec{s}(\eta, \omega)$ with an inside-outside function $f = f(x, y, z)$, the translated and rotated SE solid $\hat{\vec{s}}$ is given by

$$\hat{\vec{s}} = R_\theta R_\phi R_\psi \vec{s} + \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix}, \tag{8}$$

where θ, ϕ, and ψ are the Euler angles expressing pitch, yaw and roll respectively, $R_\theta R_\phi R_\psi$ are the associated rotation matrices and t_x, t_y, t_z are the translation in the x, y, and z directions respectively. The definition of the rotation matrices and Euler angles can be found in many standard texts on computer graphics, e.g., [Rogers and Adams 76], as well as in the computer vision literature e.g., see [Tsai 86], or [Bolle and Murthy 86].

Given the data for a general SE the system minimizes:

$$EOF_1(\epsilon_1, \epsilon_2, a_x, a_y, a_z, t_x, t_y, t_z, \theta, \phi, \psi) = \\ \textstyle\sum_{i=1\ldots n} \left(1 - \hat{f}(x_i, y_i, z_i, \epsilon_1, \epsilon_2, a_x, a_y, a_z, t_x, t_y, t_z, \theta, \phi, \psi)\right)^2 . \tag{9}$$

where x_i, y_i, z_i are the given data points from a single viewpoint, and where $\hat{f}(x_i, y_i, z_i, \epsilon_1, \epsilon_2, a_x, a_y, a_z, t_x, t_y, t_z, \theta, \phi, \psi) = f(\hat{x}, \hat{y}, \hat{z})$ and

$$\begin{bmatrix} \hat{x} \\ \hat{y} \\ \hat{z} \end{bmatrix} = R^\top \begin{bmatrix} x - t_x \\ y - t_y \\ z - t_z \end{bmatrix} .$$

where (10)

$$R^\top = \begin{bmatrix} C_\theta C_\phi & S_\psi S_\theta C_\phi - C_\psi S_\phi & C_\psi S_\theta C_\phi + S_\psi S_\phi \\ C_\theta S_\phi & S_\psi S_\theta S_\phi + C_\psi C_\phi & S_\psi S_\theta C_\phi - C_\psi S_\phi \\ -S_\theta & S_\psi C_\theta & C_\psi C_\theta \end{bmatrix}$$

To employ the Gauss-Newton iteration, the system must compute the Jacobian of the transformation, and thus also needs the partial derivatives of $\hat{f}(x, y, z)$ with respect to the 11 parameters, (5 shape, 3 rotation and 3 translation), which were obtained symbolically.

The initial implementation of the system recovered only the 5 shape parameters. Using synthetic data with up to 10% uniformly distributed noise, the system could start the minimization procedure from a canonical position (all 5 parameters = 1) and in most cases was still able to recover the underlying surface within the error of the data. Moreover, the convergence was generally quick, requiring < 15 iterations. These results are reported because, if though some other means one was able to compute the local coordinate system of the SE (or out very good bounds on it), the stability of the algorithm would be greatly increased.

When the system was extended to handle 11 parameters (i.e. 5 shape and 6 position/orientation), things became more complicated. For most of the examples presented, even when presented with very poor estimates for starting values, the system was quickly able to find an SE that had low "error of fit". Unfortunately, the solutions proposed by the system often seemed to be nonintuitive. However, when examined closely, many of these solutions proved themselves to be reasonable (though not the most reasonable) interpretations of the data. The differences could generally be traced to the nonintutative behavior of error-of-fit measure. For example, when presented with the range data (from the Utah range database, [UTAH 85]) for a coke can minus the concave portion of the bottom end, the system initially proposed a solution with $EOF_1 = .0078$. However, the parameters described an object that is slightly beveled along its long axis, and rather round in the other direction. The length of the object was far larger then expected. The exaggerated length was caused by two factors:

1. the model is assumed to be intersected with a negative ellipsoid at the top, and cut by the ground plane at the bottom, and

2. the "error of fit" measure used by the system is an underestimate, and biased toward larger objects, see Section 3 for further discussion of this problem.

When examined closely, the proposed object (assuming the volume terminates when it intersects another object in the scene) does seem to be a reasonable fit to the data, but still a cylinder seems intuitively to be a better fit. If forced to look for a cylindrical object, the system finds an SE with EOF_1 of .17. Oddly enough, the system converged to a can more readily if data from the negative ellipsoid was not removed. As reported in the next section, the use of a different error-of-fit measure resulted in better recovery.

Currently, the system obtains initial estimates of the translation parameters from the centroids of the original data and derives bounds from sensor information and overall data (maximum variation in any data). However, there are many problems with these estimates especially if the number of data points is small or if the system is only given a partial view of an object. These estimates are obviously much better if multiple views of the object are available. The estimates also assume that the data is segmented, an assumption with which the authors feel particularly uncomfortable. The system derives estimates of rotation angles and length scales from moments of inertia and bounds on the length parameters from sensor information and overall data. Because the moments of inertia require second order moments, the estimates are plagued with more difficulties than the estimates of the translation parameters. The estimates of both the rotation and length scales are very poor if an object (after segmentation) is the result of boolean combinations.

The system also estimates bounds on the maximum values for each parameter, deriving these estimates from knowledge of the "sensor" and the data values. If during the minimization process, any parameter attempts to stray beyond its allowed boundary, the system stochastically "pushes" it back toward its initial value.

4.1 A Few Examples from Our Current System

The first example in figure 13, is a synthetic SE with noisy synthetic data from multiple views. The actual parameters of the superquadric are (1.59, .39, 1, 2, 3, 1.5, 2.5, 3.5, .1, .1, .1).[5] The noise in the underlying object was uniformly distributed over the interval [-.15, +.15] and then added to the z (depth) value of a point. The 1000 data points were randomly distributed on the surface before noise was added. The system recovered an SE with parameters (1.8, .3, 1, 2, 3.58, 1.49, 2.5, 3.47, .079, .09, .101). The error (as measured by EOF_1) of the reconstruction wasf .079. The system required 7 iterations from the initial values to find the solution.

Figure 14, is the recovery of the same synthetic SE as in example 1. However, this time this system was given 1000 data points from one view of the object. Under these conditions, the recovered parameters were (2.09, .67, 1, 1.94, 3.36, 1.43, 2.53, 4.3, .075, .07, .08). The error (as measured by EOF_1) of the reconstruction was .169. Unsurprisingly, the reconstruction from multiple views is superior.

The final three examples presented, show the fitting of actual range data from the Utah range database. Figure 15 shows the elliptical indentation on the bottom of a soda can. The object which is defined by 590 data points. The system recovered the parameters (1.29, .955, .939, .930, .277, -.096, -1.55, 2.07, -.05,0,0), and had $EOF_1 = .0293$. Figure 16 shows a quasi-spherical object which is defined by 859 data points. The system recovered the parameters (.994, .951, 1.19, 1.13, 1.13, .4729, 1.437, -1.457,-.05, -.04, 0) and $EOF_1 = .021$. Figure 17 shows the cylindrical portion of a soda can defined by 1645 data points. When using the above described estimations techniques, the system recovered the parameters (2.0, .88, 1, 1.3, 19.59, -.19, -1.69, -.834, -.04,.05, 0) and with $EOF_1 = .0079$. When the inside-outside function was modified to include an extra multiplicative factor of $a_x a_y a_z$, the result was the bottom object in figure 17.

5 Conclusions and Future Direction

The main result in this paper is that there are problems and biases with currently used error-of-fit measures, as well as with two newly proposed measures. Still, the newly proposed measures often fared better than currently used measures

The analysis also suggests that even better measures await definition, especially error of fit measures which make use of knowledge of a sensor error model (direction).

The paper also summarizes results which demonstrate that even with a poor EOF measure the system can recognize both positive and negative superquadrics from depth data on only part of the surface. (The latter an important consideration if SE's are to be used with CSG operations.)

Future plans for the system also include extensions to incorporate surface derivative information. This will be accomplished by minimizing a sum with (some variant of) both the inside-outside function and a differentiated form of the inside-outside function.

Of course, one of the most important avenues for future research will be attacking the segmentation problem, our current plans are to attempt at least two approaches: pure growing of superquadrics from small data patches, and a skeletonization (to find axis) followed by both growing and splitting of superquadric solids.

A final avenues of research will be in using SE's for integration of multi-sensor (possible multi-modal) information, including some model of sensing errors. Part of this work will undoubtedly look into the use of edges in an intensity image /range image as

[5] In this example and hereafter the parameters will always be presented as a 11 vector with association implied by order given by $\epsilon_1, \epsilon_2, a_x, a_y, a_z, t_x, t_y, t_z, \theta, \phi, \psi$

they relate to limb-equations.

Acknowledgments

This work was supported in part by Darpa Contract #N00039-84-C-0165. The authors are grateful to Alex Pentland, Frank Solina and R. Bajcsy for their discussions on this subject. An special thanks to Alex Pentland (and SRI) for allowing us to use Supersketch, which (with modifications) was used to generate the renderings of SE's in the paper.

References

[Bajcsy and Solina 87a] R. Bajcsy and F. Solina. *Range image interpretation of Mail pieces with Superquadrics.* Technical Report MS-CIS-87-18, GRASP LAB 98, University of Pennsylvania, March 1987.

[Bajcsy and Solina 87b] R. Bajcsy and F. Solina. Three dimensional object representation revisted. In *Proceedings of the IEEE Computer Society International Conference on Computer Vision*, pages 231–240, IEEE, June 1987.

[Barr 81] A. H. Barr. Superquadrics and angle preserving transformations. *IEEE Computer Graphics and Applications*, 1:11–23, January 1981.

[Bolle and Murthy 86] R.M. Bolle and S.S. Murthy. *Curvature extraction from approximations to image or range data.* Technical Report, IBM Research, AI Systems Group, May 1986. IBM Thomas J. Watson Research Center.

[Bookstein 79] F. L. Bookstein. Fitting conic sections to scattered data. *Computer Vision, Graphics, and Image Processing*, 9:56–71, 1979.

[Boult and Gross 87a] T.E. Boult and A.D. Gross. Recovery of superquadrics from 3d data. In *Proceedings of the SPIE Conference on Intelligent Robots and Computer Vision: Sixth in a Series*, SPIE, November 1987. to appear, Order # 848-55.

[Boult and Gross 87b] T.E. Boult and A.D. Gross. Recovery of superquadrics from depth information. In *Proceedings of the AAAI Workshop on Spatial-Reasoning and Multisensor Integration*, AAAI, October 1987.

[Brooks 85] R.A. Brooks. Model based 3-d interpretation of 2-d images. In A. Pentland, editor, *From Pixels to Predicated*, Ablex Publishing Co., Norwood, N.J., 1985.

[Brooks and Binford 80] R.A. Brooks and T. O. Binford. Representing and reasoning about specific scenes. In *Proceedings of the DARPA Image Understanding Workshop*, pages 95–103, DARPA, April 1980.

[Gardiner 65] M. Gardiner. The superellipse: a curve that lies between the ellipse and the rectangle. *Scientific American*, September 1965.

[Hageman and Young 70] L.A. Hageman and D.M. Young. *Applied Iterative Methods*. Academic Press, NYC, NY, 1970.

[Pentland 86a] A. Pentland. *Recognition by Parts*. Technical Report 406, SRI International, December 1986.

[Pentland 86b] A. Pentland. Towards an ideal 3-d cad system. In *Proceedings of the SPIE Conference on Machine Vision and the Man-Machine Interface*, SPIE, Janurary 1986. Order # 758-20.

[Pentland 87] A. P. Pentland. Recognition by parts. In *Proceedings of the IEEE Computer Society International Conference on Computer Vision*, pages 612–620, IEEE, June 1987.

[Ponce, Chelberg and Mann 87] J. Ponce, D. Chelberg, and W. Mann. Invariant properties of the projection of straight homogenous generalized cylinders. In *Proceedings of the IEEE Computer Society International Conference on Computer Vision*, pages 631–635, IEEE, June 1987.

[Pratt 87] V. Pratt. Direct least-squares fitting of algebraic surfaces. In *Proceedings of SIGGRAPH '87: Computer Graphics 21:4*, pages 145–152, ACM, July 1987.

[Rao and Nevatia 86] K. Rao and R. Nevatia. Generalized cone descriptions from sparse 3-d data. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 256–263, IEEE, 1986.

[Rogers and Adams 76] D.F. Rogers and J.A. Adams. *Mathematical Elements for Computer Graphics.* McGraw-Hill, New York, 1976.

[Sampson 82] P.D. Sampson. Fitting conic sections to "very scattered" data: an iterative refinment o. the bookstein algorithm. *Computer Vision, Graphics, and Image Processing*, 18:97–108, 1982.

[Solina 87] F. Solina. *Shape recovery and segmentatin with deformable part models.* PhD thesis, University of Pennsylvania, Department of Computer Science., 1987. available as Tech. Report MS-CIS-87-111.

[Tsai 86] R.Y. Tsai. An efficient and accurate camera calibration technique for 3d machine vision. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 364–374, June 1986.

[Turner 74] K.J. Turner. *Computer perception of curved objects using a television camera.* PhD thesis, Dept. of Machine Intelligence, University of Edinburgh, 1974.

[UTAH 85] utah. *The University of Utah range database manual.* Technical Report , University of Utah, 1985.

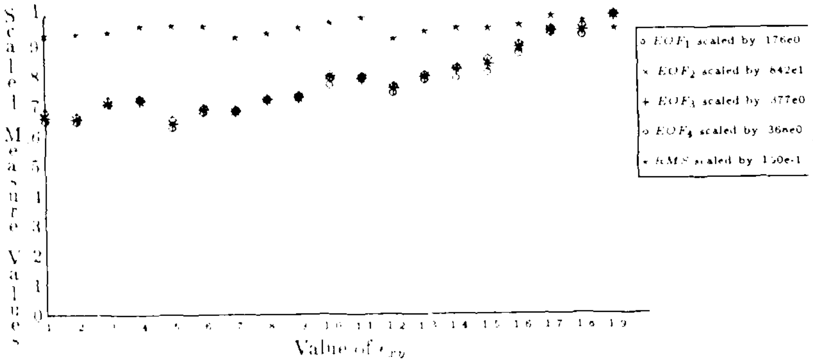


Figure 4: Bias experiments for ϵ_{xy} with $\epsilon_{xz} = .1$ $a_x = 1$, $a_y = 10$ and $a_z = 50$

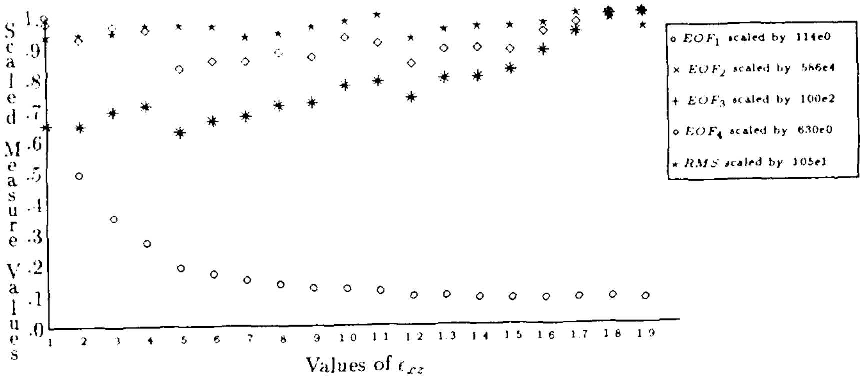


Figure 5: Bias experiments for ϵ_{xz} with with $\epsilon_{xy} = .1$, $a_x = 70, a_y = 70$ and $a_z = 70$

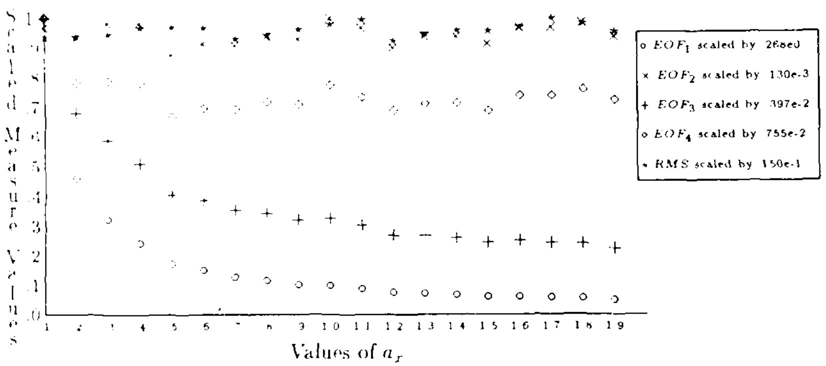


Figure 6: Bias experiments for a_x with $\epsilon_{xz} = .5$, $\epsilon_{xy} = .5$, $a_y = .1$ and $a_z = .1$

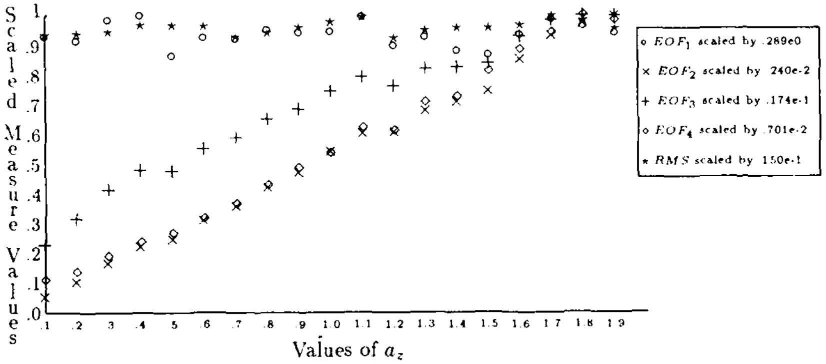


Figure 7: Bias experiments for a_z with $\epsilon_{xz} = .5$, $\epsilon_{xy} = .5$, $a_x = .1$ and $a_y = .1$.

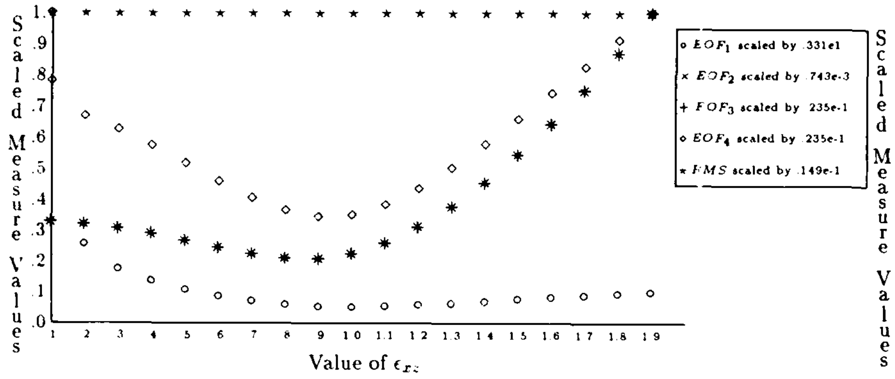


Figure 8: Cross section of error surface as one varies ϵ_{xz} with correct parameters: $\epsilon_{xz} = 1, \epsilon_{xy} = 1, a_x = .1, a_y = .1$ and $a_z = .1$

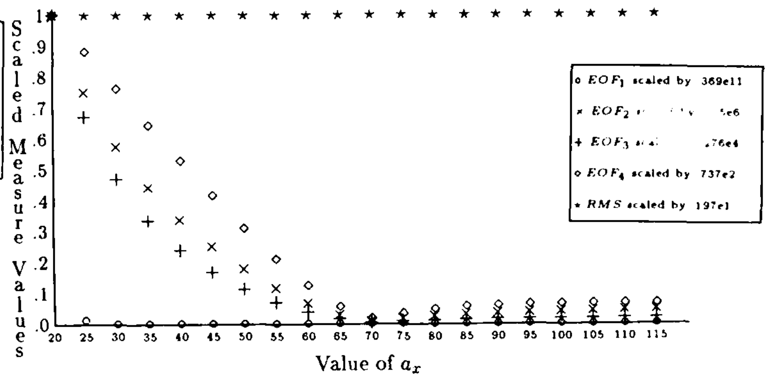


Figure 9: Cross section of error surface as one varies a_x with correct parameters: $\epsilon_{xz} = .1, \epsilon_{xy} = .1$, $a_x = 70, a_y = 70$ and $a_z = 70$

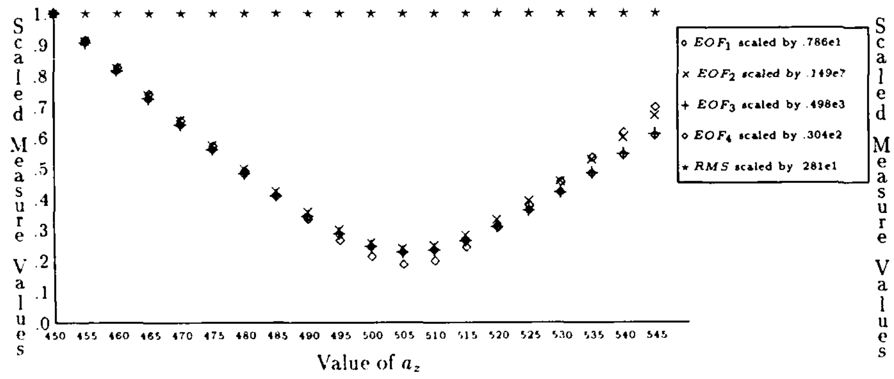


Figure
10: Cross section of error surface as one varies a_z with correct parameter: $\epsilon_{xz} = 2, \epsilon_{xy} = .1$, $a_x = 100, a_y = 200$ and $a_z = 500$

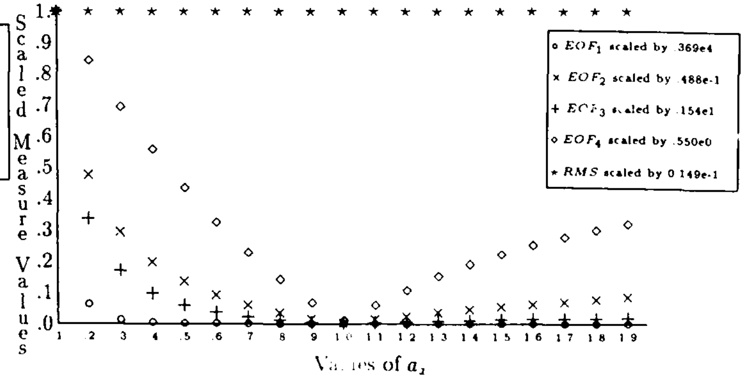


Figure 11: Cross section of error surface as one varies a_x with $\epsilon_{xz} = .5, \epsilon_{xy} = .5$, $a_x = 1.0, a_y = .1$ and $a_z = .1$

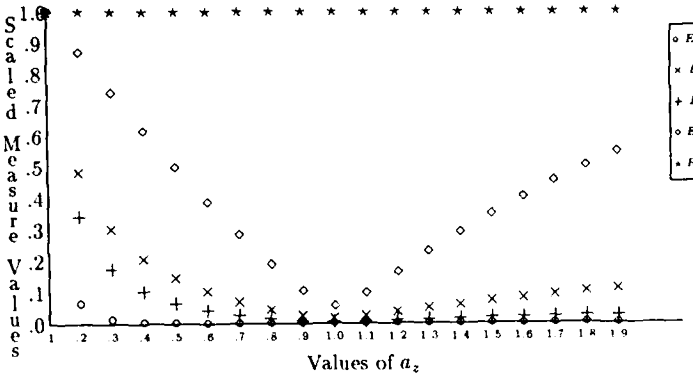


Figure 12: Cross section of error surface as one varies a_z with $\epsilon_{xz} = .5, \epsilon_{xy} = .5$, $a_x = .1, a_y = .1$ and $a_z = 1.0$

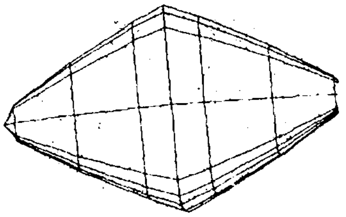


Figure 13: Example recovering 11 parameters using noisy synthetic data from single views

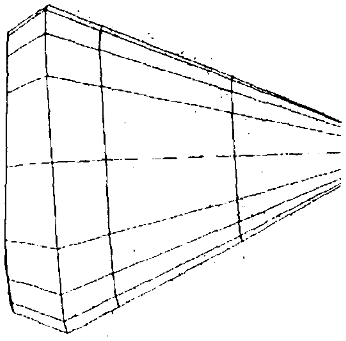


Figure 14: Example recovering 11 parameters using noisy synthetic data from single views

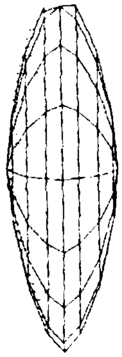


Figure 15: Reconstruction of a negative ellipsoid from real range data

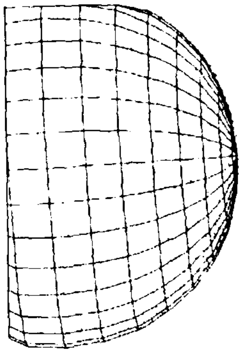


Figure 16: Reconstructed sphere from actual range data

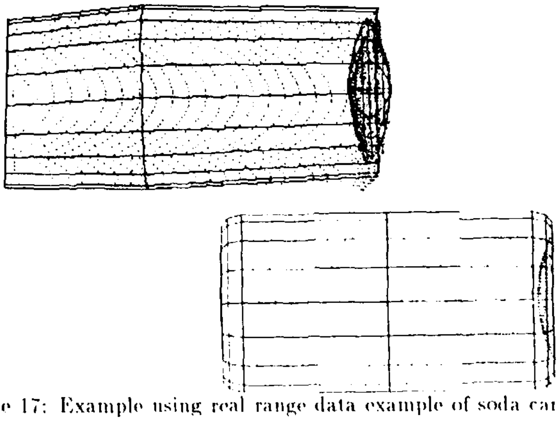


Figure 17: Example using real range data example of soda can, best fit surface with standard inside-outside function.(top) and with function multiplied by $a_x a_y a_z$ (bottom).

Straight Homogeneous Generalized Cylinders: Differential Geometry And Uniqueness Results*

Jean Ponce

Robotics Laboratory
Department of Computer Science
Stanford University
Stanford, California 94305

Abstract: In this paper, we study the differential geometry of straight homogeneous generalized cylinders (SHGC's). We derive a necessary and sufficient condition that a SHGC must verify to parameterize a regular surface, compute the Gaussian curvature of a regular SHGC, and prove that the parabolic lines of a SHGC are either meridians or parallels. Using these results, we address in the second part of the paper the following problem: Under which conditions can a given surface have several descriptions by SHGC's? We prove several new results. In particular, we prove that two SHGC's with the same cross-section plane and axis direction are necessarily deduced from each other through inverse scalings of their cross-sections and sweeping rule curve. We extend Shafer's pivot and slant theorems. Finally, we prove that a surface with at least two parabolic lines has at most three different SHGC descriptions, and that a surface with at least four parabolic lines has at most a unique SHGC description.

Introduction

A *generalized cylinder* [Binford, 1971] is the solid obtained by sweeping a planar region, its *cross-section*, along a space curve, its *axis*, or *spine*. The axis is not necessary straight, or even planar; the cross-section is not necessarily circular, or even constant; its deformation is governed by a *sweeping rule*.

Generalized cylinders have been extensively used to represent three-dimensional objects in computer vision [Agin, 1972], [Brooks, 1981], [Nevatia, 1982], [Ponce et al., 1°87], [Binford et al., 1°87], [Horaud and Brady, 1987]. The most successful vision system to date using generalized cylinders as its primary representation for three-dimensional objects is probably *Acronym* [Brooks, 1981]. The generalized cylinders used in Acronym are very simple (circular or regular polygonal cross-section, linear or bi-linear sweeping rule). To develop a vision system using a richer class of generalized cylinders, it is necessary to understand the geometry of these primitives, whether a same object may be described by different primitives or has a unique description, and how these primitives project into images.

Shafer, in his pioneering work [Shafer, 1985], addressed these three problems for the case of *straight homogeneous generalized cylinders* (SHGC's, see [Shafer, 1985]), obtained by scaling an arbitrary cross-section along a straight axis. He derived expressions for the surface and the normal of a SHGC and proved several uniqueness results and properties of the contours of SHGC's. This paper is one in a series of articles that study the geometry of straight homogeneous generalized cylinders in an effort to extend Shafer's work. In [Ponce and Chelberg, 1987] and [Ponce, Chelberg, and Mann, 1987], we have studied the contours of SHGC's, and proved new properties of these contours which are invariant with respect to the viewing direction.

In this paper, we study the differential geometry of SHGC's, and use our conclusions to derive new uniqueness results. In section 1, we prove that the surface of a general SHGC is not always regular (among other things, it does not always have a well-defined tangent plane everywhere), and derive necessary and sufficient conditions of regularity. We derive in section 2 an expression for the Gaussian curvature of a SHGC and prove that the parabolic lines of a SHGC are always either meridians or parallels. In the following section, we define, following Shafer, classes of equivalent SHGC's, which trivially define the same surface. We prove the new converse result that if a surface is described by two SHGC's with the same axis and cross-section plane,

*Support for this work was provided in part by the Air Force Office of Scientific Research under contract F33615-85-C-5106 and by the Advanced Research Projects Agency of the Department of Defense under Knowledge Based Vision ADS subcontract S1093 S-1.

then these two SHGC's are equivalent.

In section 4, we use the result of section 1 on SHGC's parabolic lines to prove a simple uniqueness result, stating that a surface described by a SHGC with two parabolic meridians and two parabolic parallels does not have any other non-equivalent SHGC description. In section 5, we extend Shafer's slant and pivot theorems. Relaxing some of his assumptions (intersecting axes, same radius functions), we prove that for a non-linear SHGC, the direction of the axis is determined by the direction of the cross-section, and that the direction of the cross-section is determined by the direction of the axis. Finally, in section 6, we give our main uniqueness results. A surface with at least two parabolic lines has at most one SHGC description if these lines are parallel, and at most three if they intersect. A closed surface with at least one concave point has at most three SHGC descriptions. A surface with at least four parabolic lines has at most one SHGC description.

The proofs of our results are in general simple, but rather technical. The most technical proofs are given in the appendix, and only the "geometric" proofs of the uniqueness results are given in the main body of the paper.

1. Straight homogeneous generalized cylinders

In this section, we first give a more precise definition of straight homogeneous generalized cylinders, and then study their differential geometry. We introduce the notion of *regular* SHGC and give a necessary and sufficient condition or regularity. Informally, a straight homogeneous generalized cylinder (SHGC, see [Shafer,1985]) is obtained by scaling a planar cross-section along a straight axis. Let us give first a more formal definition.

Definition 1.1: *Let $\mathbf{x}_0$ be a point of R^3, $P = (\mathbf{i}, \mathbf{j}, \mathbf{k})$ an orthonormal basis of R^3, $\mathbf{a} = (p, q)$ a vector of R^2, and S and C two simple regular planar curves respectively parameterized by $(z, r) : I \subset R \rightarrow R^2$ and $(x, y) : J \subset R \rightarrow R^2$, where I and J are open intervals of R. The parameterized surface $\mathbf{x} : I \times J \rightarrow R^3$ defined in the affine coordinate system $(\mathbf{x}_0, \mathbf{i}, \mathbf{j}, \mathbf{k})$ by*

$$\mathbf{x}(s,t) = r(s) \begin{pmatrix} x(t) \\ y(t) \\ 0 \end{pmatrix} + z(s) \begin{pmatrix} p \\ q \\ 1 \end{pmatrix}, \qquad (1.1)$$

is the straight homogeneous generalized cylinder associated with the five-tuple $(\mathbf{x}_0, P, \mathbf{a}, S, C)$. The point $\mathbf{x}_0$ is the origin of the SHGC, P is (the coordinate system associated with) its cross-section plane, and $\mathbf{a}$ is the direction of its axis. The parameterized curves C and S are respectively the reference cross-section and the sweeping rule of the SHGC.

Notice that this formulation is very general. The shape of the reference cross-section is completely arbitrary. Similarly, the shape of the sweeping rule curve is arbitrary. In particular, r is not necessarily a function of z. The axis is not necessarily orthogonal to the cross-section plane, the only restriction is that it does not lie in this plane. Most authors have used a more restricted definition of SHGC's. Marr [1977] mostly restricts his study of SHGC's (which he calls *generalized cones*) to SHGC's with convex cross-sections. In previous papers [Ponce and Chelberg, 1987], [Ponce, Chelberg, and Mann, 1987], we mainly restricted our study to star-shaped cross-sections. Shafer [1985] considers an arbitrary cross-section, but assumes as the others that the scaling is given as a function of z. This is a restricting assumption, as for example, a torus can only be described as a SHGC when both r and z are parameterized.

Let us now examine under which conditions the surface associated to a SHGC is regular, i.e., in particular, it is smooth, has no self-intersections, and has a well-defined tangent plane at each of its points (see [do Carmo, *1976*]). *This is important* as most concepts of differential geometry only apply to regular surfaces. For example, it is well known (see [do Carmo, 1976]) that a solid of revolution, defined as above with $x = \cos t, y = \sin t$, is regular as long as r is strictly positive. We are going in all this paper to make the following assumptions (this is similar to Shafer's definition of "non-degenerate" SHGC's).

Assumptions: To avoid degenerate cases, we assume in all this paper that: (a.1): r is strictly positive; (a.2): the sweeping rule curve is not an horizontal line segment (i.e., z' is not identically 0); and (a.3): the cross-section curve is not a straight line segment whose supporting line goes through the origin (i.e., $x'y - y'x$ is not identically 0).

Under these assumptions, we have the following result.

Proposition 1.1: *The surface associated with a SHGC is regular iff at least one of the two following conditions is verified: (a): $\forall s \in I, \quad z'(s) \neq 0$, (b): $\forall t \in J, \quad x'(t)y(t) - y'(t)x(t) \neq 0$.*

The detailed proof can be found in the appendix. In particular, for a regular SHGC, the tangent plane is defined everywhere, and the (non-normalized) normal

$\mathbf{n}$ to the surface at a point $\mathbf{x}(s,t)$ is given by

$$\mathbf{n} = \mathbf{x}_t \times \mathbf{x}_s = rr'(x'y - y'x)\begin{pmatrix}0\\0\\1\end{pmatrix} + rz'\begin{pmatrix}y'\\-x'\\x'q - y'p\end{pmatrix}. \tag{1.2}$$

We prove in the appendix that if both conditions (a) and (b) are false, then the surface is not regular because there is at least one point where the normal is not defined. In fact, it is easy to show that if z and x/y are both extremal, then the surface has self-intersections. A geometric interpretation of the previous proposition is given by the following immediate corollary.

Proposition 1.2: *The surface associated to a SHGC is regular iff the sweeping curve can globally be parameterized as a function r of z, or the reference cross-section can globally be parameterized in polar coordinates as a function ρ of θ.*

Notice that a closed cross-section can be parameterized in polar coordinates iff it is star-shaped. There exist open curves which can be parameterized in polar coordinates but are such that the line between a point and the origin intersects the curve several times (e.g., an exponential spiral).

The above proposition has practical consequences. Non star-shaped SHGC's and folding SHGC's (i.e., SHGC's for which r cannot be written as a function of z) are worth considering for vision and modelling purposes, as they describe a wide class of objects However, SHGC's which have both a folding sweeping rule curve and a non-star-shaped cross-section are ill-defined objects, and there is no point including them in a vision or a geometric modelling system (e.g., as the one described in [Ponce et al., 1987]).

2. Intrinsic geometry of SHGC's

We now turn to the intrinsic geometry of the surfaces associated with straight homogeneous generalized cylinders. We compute the Gaussian curvature of a regular SHGC, and prove that the parabolic lines of a SHGC are always meridians or parallels.

Proposition 2.1: *The parabolic lines of a SHGC are its meridians which correspond to points of its reference cross-section where the tangent is aligned with the vector (x, y) or the curvature is zero, and its parallels which correspond to points of its sweeping rule curve where the tangent is parallel to the cross-section plane or the curvature is zero.*

The proof of the proposition can be found in the appendix. In particular, we prove that the Gaussian curvature is

$$K = \frac{r^3}{|\mathbf{n}|^4}(z''r' - r''z')(x''y' - y''x')z'(x'y - y'x). \tag{2.1}$$

The proposition follows easily from this expression.

Notice that a regular SHGC cannot have both parabolic parallels where $r'(s) = 0$ and parabolic meridians where $(x'y - y'x)(t) = 0$. However, it may have at the same time parabolic parallels corresponding to zeros of curvature of the sweeping rule and parabolic meridians corresponding to zeros of curvature of the reference cross-section. Related work can be found in [Brady et al., 1985]. They prove that the only meridians and parallels of a SHGC which are lines of curvature are the extremal ones, called flutings and skeletons (see also [Marr, 1977]). As shown in [Ponce, Chelberg, and Mann, 1987], this result is only valid for right SHGC's.

A first consequence of this expression of the Gaussian curvature is that the zeros of curvature of the contours of SHGC's correspond to points where the curvature of the sweeping rule or the cross-section is 0, or where $z'(s) =$ or $(x'y - y'x)(t) = 0$. This is due to a theorem by Koenderink [1984], which relates the curvature of the contours of an object to its Gaussian curvature. This result can also be proved directly. See [Ponce, Chelberg, and Mann, 1987] for details.

In this paper, the most important consequence of this proposition is that it can be used to prove that surfaces with parabolic lines have a (small) finite number of possible SHGC descriptions (see section 4 and 6).

3. Equivalent SHGC's

Assumptions (a.1) to (a.3) were sufficient to define straight homogeneous generalized cylinders. We are now going to look at alternate descriptions of objects by SHGC's, and we will need the two additional assumptions.

Assumptions: We suppose from now on that: (a.4): the sweeping rule curve of a SHGC does not contain horizontal line segments (i.e., there is no non-empty open interval such that z' is zero on that interval), and (a.5): the reference cross-section does not contain line segments going through the origin (i.e., there is no non-empty open interval such that $x'y - y'x$ is zero on that interval).

Notice that for a given regular SHGC, we just need one of these assumptions, as the other one will be automatically verified. A straight homogeneous generalized cylinder, as defined earlier, is a parameterization of a surface and not the surface itself. In particular, two different SHGC's may parameterize the same surface. Following Shafer [1985], we now introduce the notion of *equivalent SHGC's.*

Definition 3.1: *Two SHGC's are said to be equivalent if they can be deduced from each other through any sequence of the following transformations: changing the parameterization of the sweeping rule or the reference cross-section; translating the origin along the axis by a given constant while translating z by the opposite of that constant; rotating the vectors* $(\mathbf{i}, \mathbf{j})$ *around the vector* $\mathbf{k}$ *by a given angle while rotating* (x, y) *and* $\mathbf{a}$ *by the opposite angle; changing the orientation of the cross-section plane, the axis, and the sweeping rule and cross-section curves; scaling the sweeping rule by a non-zero constant while scaling the reference cross-section by the inverse of that constant.*

Trivially, two equivalent SHGC's parameterize the same surface, as already remarked by Shafer. We can in fact prove the converse result.

Proposition 3.1: *The surfaces associated with two SHGC's with the same (non-oriented) cross-section plane and (non-oriented) axis are the same iff the two SHGC's are equivalent.*

Notice that, trivially, the relation defined by the equivalence of two SHGC's is a relation of equivalence. From now on, a SHGC will be a class of equivalence of this relation, represented by any member of the class. This allows us to talk about the cross-section plane, the axis, the meridians and parallels associated to a SHGC independently of the coordinate system and orientation chosen to describe them.

4. A simple sufficient uniqueness condition

We now have all the tools to prove a first, simple uniqueness result. We are going to use the (immediate) following properties of the parallels and meridians of a SHGC.

Properties: The sweeping rule (respectively reference cross-section) of a SHGC is not linear iff there exists a non-linear meridian (respectively parallel). A non-linear meridian (respectively parallel) uniquely determines a plane in which it lies. The planes of any two non-linear meridians intersect on the axis. The planes of a non-linear meridian and a non-linear parallel intersect.

Proposition 4.1: *If a surface admits a parameterization by a SHGC whose sweeping rule has two zeros of curvature (corresponding to different values of z) and the cross-section is not linear, then the parabolic parallels corresponding to these zeros of curvature are parallels in any othe, description of the surface by a SHGC.*

Proof of the proposition: The two zeros of curvature of the sweeping rule correspond to two parabolic parallels P_1 and P_2. In any other description, P_1 and P_2 are either parallels or meridians. As the cross-section is not linear, P_1 and P_2 are themselves not linear and uniquely determine two parallel planes. P_1 and P_2 cannot be both meridians as they are not linear and their planes don't intersect (these planes are parallel and don't coincide as the zeros of curvature correspond to different values of z).

Suppose that one of them is a parallel and the other one is a meridian. Then in this description, neither the cross section nor the sweeping rule is linear, so the planes of P_1 and P_2 should intersect, which contradicts the hypothesis. It follows that P_1 and P_2 are parallels in any descriptions. ■

Proposition 4.2: *If a surface admits a SHGC description such that the sweeping rule curve is not linear and has at least two zeros of curvature (corresponding to different values of z) and the reference cross-section is not linear and has at least two (non-diametrically opposed) zeros of curvature, then the surface admits no other different SHGC description.*

Proof of the proposition: From the previous proposition, the two parabolic parallels are parallels in any description. The two parabolic meridians M_1 and M_2 corresponding to the zeros of curvature of the reference cross-section are meridians or parallels in any description. As M_1 and M_2 are not linear, they determine two planes, which intersect the planes of the two parabolic parallels. Therefore, M_1 and M_2 cannot be parallels and are meridians in any description. As there are not linear, their planes intersect on the axis. Both the cross-section plane and the axis are given, and using Proposition (3.1), the proposition is proved. ■

Notice that the above result is relatively weak. However, a non-convex closed cross-section has at least two zeros of curvature. Similarly, is the surface of a SHGC is closed and its sweeping rule is non-convex, then it also has at least two zeros of curvature. This fairly general class of surfaces admits a unique SHGC description. To prove stronger uniqueness results, we are now going to extend some results due to Shafer.

5. Generalized pivot and slant theorems

In this section, we prove generalized versions of Shafer's pivot and slant theorems [Shafer, 1985].

Pivot theorem: *A non-degenerate SHGC can be described as another SHGC with a different axis, and the same cross section planes, iff it is linear.*

This theorem has been proved under the assumption that the two parameterizations have intersecting axes.

Slant theorem: *A non-degenerate oblique SHGC has an equivalent right SHGC iff the radius function is linear.*

This theorem has been proved under the assumption that the same scaling function r is to be used for both parameterizations. It has been extended by Roberts [1985], to relax this assumption. Both theorems have been proved under the additional assumption that r is a function of z. In this section, we are going to give more general statements, valid for general SHGC's, without the above restrictions.

Proposition 5.1: *The surface associated with a non linear SHGC cannot be described by another SHGC with a different axis, and the same cross-section plane.*

Proposition 5.2: *The surface associated with a non linear SHGC cannot be described by another SHGC with the same axis, and a different cross-section plane.*

A detailed proof of these two propositions can be found in the appendix. The important consequence of these propositions is that for a given surface, and a given cross-section plane (respectively a given axis), there is at most one possible description by a non-linear SHGC (class of equivalence).

6. Main uniqueness results

Proposition 6.1: *A surface with at least two non-linear parabolic lines has at most a unique description as a SHGC if these lines are parallel, and at most three different SHGC descriptions if they are not.*

Proof of the proposition: Let us suppose that a surface has two non-linear parabolic lines L_1 and L_2. If they are parallel to each other, then they are parallels in any description of that surface by a SHGC. Using Proposition (5.1), this also determines the axis. If L_1 and L_2 are not parallel to each other, there are three possible cases. If L_1 and L_2 are meridians, the intersection of their planes determines the axis and therefore also the cross-sections of any SHGC description (Proposition (5.2)). If L_1 is a parallel and L_2 is a meridian, then L_1 determines the cross-section plane and therefore the axis (Proposition 5.1 again). In the remaining case, L_2 is a parallel and L_1 is a meridian. L_2 determines the cross-section plane, and a corresponding third potential description of the surface as a SHGC. ■

A corollary of this result is that:

Proposition 6.2: *A closed bounded surface with at least one concave point has at most one SHGC description.*

Proof of the proposition: A closed bounded surface has at least one convex point, where both principal curvatures are strictly positive and the Gaussian curvature is strictly positive (e.g., [do Carmo, pp. 172]). By hypothesis, the surface also has a concave point, where both principal curvatures are strictly negative and the Gaussian curvature is strictly positive. The surface has therefore two parabolic lines, separating one hyperbolic region and two elliptic regions. The result follows. ■

Proposition 6.3: *A surface with at least four non-linear parabolic lines has at most one SHGC description.*

Proof of the proposition: Suppose now that the surface has at least four non-linear parabolic lines. If at least two of these lines lie in parallel planes, we have just proved that the surface has at most one SHGC parameterization. In the remaining case, suppose that the surface can be parameterized by a SHGC. In this description, at most one of the lines is a parallel, and the three others lines L_1,L_2,L_3 intersect on the axis.

Suppose then that the surface admits an other parameterization by a SHGC. In this description, at most one of the L_i's can be a parallel, as any two of these lines intersect. The two other lines are meridians, and their intersection gives the axis of the second description. This axis is the same as in the first description, and the result is proved. ■

In particular, consider the case of an ellipsoid of revolution. If we create one concave point by pressing on one of its ends, but keeping it a solid of revolution, the above results show that the resulting object has a unique SHGC description, although the figure is still radially symmetric.

Conclusion

We have proved new results concerning the differential geometry of straight homogeneous generalized cylinders, and used them to derive new uniqueness results for these objects. We plan to use similar methods to derive similar results for a wider class of generalized cylinders. Some problems that we will address are: When is the surface of a generalized cylinder with a curved axis and a circular cross-section regular? Can we prove uniqueness results for these primitives? We already know for example that a torus can be described by a SHGC, but also by a generalized cylinder with a circular axis and a constant circular cross-section. Can it be described by other generalized cylinders? An other interesting problem is, given the model of an object, to try to determine (when possible) not only whether it may have alternate representations, but also how many, and maybe generate these representations automatically.

Acknowledgements: I wish to thank Tom Binford, Ron Fearing, Glenn Healey, Dave Kriegman, and Rick Vistnes for useful discussions and comments on earlier drafts of this paper.

References

1. Agin, G.J.,"Representation and description of curved objects", (Ph.D. dissertation), AIM-273, Stanford Artificial Intelligence Lab., October 1972.

2. Binford, T.O., "Visual perception by computer", Proc. IEEE Conference on Systems and Control, Miami, December 1971.

3. Binford, T.O, Levitt, T., and Mann, W., "Bayesian inference in model-based machine vision", Proc. Workshop on Uncertainty in Artificial Intelligence, 1987.

4. Brady, M., Ponce, J., Yuille, A., and Asada, H., "Describing surfaces", Proc. 2^{nd} International Symposium on Robotics Research, Harasuja and Inoue (ed.), MIT Press, 1985.

5. Brooks, R.A., "Symbolic reasoning among 3D models and 2D images", Artificial Intelligence 17, pp. 285-348, 1981.

6. do Carmo, M.P., "**Differential geometry of curves and surfaces**", Prentice-Hall, 1976.

7. Horaud, R., and Brady, M., "On the geometric interpretation of image contours", Proc. First International Conference on Computer Vision, London, U.K., June 1987.

8. Koenderink, J.J., "What does the occluding contour tell us about solid shape", Perception, Vol. 13, 1984.

9. Marr, D., "Analysis of occluding contour", Proc. Royal Society of London, B-197, pp. 441-475, 1977.

10. Marr, D., and Nishihara, K., "Representation and recognition of the spatial orginazition of three dimensional shapes", Proc. Royal Society of London, B-200, pp. 269-294, 1978.

11. Nevatia, R., "**Machine perception**", Prentice-Hall, Englewood Cliffs, NJ, 1982.

12. Ponce, J., and Chelberg, D., "Finding the limbs and cusps of generalized cylinders", International Journal of Computer Vision, Vol. 1 No. 3, 1987.

13. Ponce, J., Chelberg, D., and Mann, W., "Invariant properties of straight homogeneous generalized cylinders and their contours", submitted to IEEE Transactions PAMI, 1987.

14. Ponce, J., Chelberg, D., Kriegman, D.J., and Mann, W., "Geometric modelling with generalized cylinders", Proc. IEEE Workshop on Computer Vision, Miami, Dec. 1987.

15. Roberts, K.R., "Equivalent descriptions of generalized cylinders", Proc. Image Understanding Workshop, Miami, December 1985.

16. Rudin, W., "Principles of mathematical analysis", McGraw-Hill, 1976.

17. Shafer, S.A.,"**Shadows and silhouettes in computer vision**", Kluwer Academic Publishers, 1985.

7. Appendix: proofs of the propositions

In this section, we give the proofs of propositions (1.1), (2.1), (3.1), (5.1), and (5.2). We use some lemmas, whose proof is simple but omitted for the sake of conciseness.

Proposition 1.1: *The surface associated with a SHGC is regular iff one of the two following conditions is verified: (a): $\forall s \in I, \quad z'(s) \neq 0$, (b): $\forall t \in J, \quad x'(t)y(t) - y'(t)x(t) \neq 0$,*

To prove this proposition, we will need the following lemma.

Lemma 7.1: *Let $f : I \rightarrow R$ be a continuous real function defined on the open interval $I \subset R$. If f is not identically 0 on I, and has at least one zero on I, then there exists a point $t_0 \in I$ such that $f(t_0) = 0$ and that there is no open interval J included in I and containing t_0 such that, for all $t \in J, f(t) = 0$.*

Proof of the proposition: To prove that the surface associated with a SHGC satisfying (a) or (b) is regular, it is sufficicent to prove (see [do Carmo, 1976, pp. 52]) that: (1): the function $\mathbf{x}$ is differentiable, (2): it is an homomorphism (i.e, it is continuous and invertible, and its inverse is continuous), and (3): $\forall(s,t) \in I \times J, \mathbf{x}_t \times \mathbf{x}_s \neq \mathbf{0}$.

Let us suppose that (a) or (b) is satisfied. (1) is trivial. We now prove (2). From now on, we drop the arguments s and t. $\mathbf{x}$ is clearly continuous and surjective. To prove that it is injective and that its inverse is continuous, let us consider a point $(X,Y,Z)^t$ on the surface, and a point (s,t) of $I \times J$ such that $\mathbf{x}(s,t) = (X,Y,Z)^t$.

Let us suppose first that the condition (a) is verified. In this case, the function z is invertible on I, and we have $s = z^{-1}(Z)$, where z^{-1} is continuous (this is a consequence of the inverse function theorem, e.g., see [Rudin, 1976, pp. 221]).

We deduce that $x(t) = (X - Zp)/r \circ z^{-1}(Z)$ and $y(t) = (Y - Zq)/r \circ z^{-1}(Z)$, where "o" denotes the composition of two functions. As $(x,y) : J \rightarrow R^2$ is a parameterization of the reference cross-section, t is a continuous function of (X,Y,Z).

We now consider the case where (b) is verified. In particular, $x(t)$ and $y(t)$ cannot both be zero in t. At a point where $Y - Zq = r(s)y(t) \neq 0$, we get $(x/y)(t) = (X - Zp)/(Y - Zq)$.

As (b) is verified, x/y is invertible in a neighborhood of this point and admits a continuous inverse (again a consequence of the inverse function theorem). We get $t = (x/y)^{-1} \circ ((X - Zp)/(Y - Zq))$. The same reasoning holds in the neighborhood of a point where $X - Zp = r(s)x(t) \neq 0$, so t is a continuous function of (X,Y,Z).

We have $r(s) = (X - Zp)/x \circ t^{-1}(X,Y,Z)$ and $z(s) = Z$. As $(z,r) : I \rightarrow R^2$ is a parameterization of the sweeping rule curve, s is a continuous function of (X,Y,Z). This completes the proof that $\mathbf{x}$ is an homomorphism.

We now prove (3), i.e., $\mathbf{x}_t \times \mathbf{x}_s \neq \mathbf{0}$. The partial derivatives of $\mathbf{x}$ with respect to s and t are respectively given by

$$\mathbf{x}_s = r' \begin{pmatrix} x \\ y \\ 0 \end{pmatrix} + z' \begin{pmatrix} p \\ q \\ 1 \end{pmatrix}, \mathbf{x}_t = r \begin{pmatrix} x' \\ y' \\ 0 \end{pmatrix}. \qquad (7.1)$$

Their cross product is given by

$$\mathbf{x}_t \times \mathbf{x}_s = rr' \begin{pmatrix} 0 \\ 0 \\ x'y - y'x \end{pmatrix} + rz' \begin{pmatrix} y' \\ -x' \\ x'q - y'p \end{pmatrix}. \quad (7.2)$$

From the previous expression and the fact that C is regular, so for all $t \in J$, $x'^2 + y'^2 \neq 0$, we deduce immediatly that (3) is satisfied.

We have proved that if (a) or (b) is satisfied, then the surface is regular. We now prove the converse proposition. Let us suppose that neither (a) nor (b) is satisfied, i.e., there exists $(u_0, v_0) \in I \times J$, such that $z'(u_0) = 0$, and $(x'y - y'x)(v_0) = 0$.

As, from assumption (a.2), z' is not identically zero, it follows form lemma (7.1) that there is a point $s_0 \in I$ such that $z'(s_0) = 0$ and there is no open interval containing s_0 such z' is identically 0 on this interval. In particular, it follows that for all $\epsilon > 0$, there exists $s_\epsilon \in I$, such that $|s_\epsilon - s| < \epsilon$ and $z'(s_\epsilon) \neq 0$. Let us define the sequence $(s_n)_{n>0}$ by $s_n = s_{\epsilon=1/n}$. The sequence $(s_n)_{n>0}$ converges towards s_0.

Similarly, using assumption (a.3) and lemma (7.1), we can define a point $t_0 \in J$ and a sequence $(t_n)_{n>0}$ such that $(x'y - y'x)(t_0) = 0$, for all $n > 0, (x'y - y'x)(t_n) \neq 0$, and the sequence $(t_n)_{n>0}$ converges towards t_0.

At a point (s_n, t_0), the normal is parallel to the constant vector $(y', -x', x'q - y'p)^t$. At a point (s_0, t_n), the normal is parallel to the constant vector $(0,0,1)^t$.

If the surface is regular, then it is locally orientable in $\mathbf{x}(s_0, t_0)$, and the normal is well-defined and continuous in a neighborood of this point (e.g., see [do Carmo, 1976, pp. 136]). In particular the sequences $(\mathbf{n}(s_n, t_0))_{n>0}$ and $(\mathbf{n}(s_0, t_n))_{n>0}$, (where $\mathbf{n}(s,t)$ denotes the normal to the surface at the point $\mathbf{x}(s,t)$) should converge toward a same direction. But these sequences are constant and parallel to independent vectors. This contradicts the hypothesis, and the result is proved. ■

Proposition 2.1:*The parabolic lines of a SHGC are its meridians which correspond to points of its reference cross-section where the tangent is aligned with the vector (x,y) or the curvature is zero, and its parallels which correspond to points of its sweeping rule curve where the tangent is parallel to the cross-section plane or the curvature is zero.*

Proof of the proposition: We compute the Gaussian curvature of a SHGC. The second partial derivatives of $\mathbf{x}$ are given by

$$\mathbf{x}_{ss} = r'' \begin{pmatrix} x \\ y \\ 0 \end{pmatrix} + z'' \begin{pmatrix} p \\ q \\ 1 \end{pmatrix}, \mathbf{x}_{st} = r' \begin{pmatrix} x' \\ y' \\ 0 \end{pmatrix},$$

$$\mathbf{x}_{tt} = r \begin{pmatrix} x'' \\ y'' \\ 0 \end{pmatrix}. \qquad (7.3)$$

The Gaussian curvature of the surface is given by

$$K = \frac{eg - f^2}{EG - F^2}, \qquad (7.4)$$

where E, F, G and e, f, g are respectively the coefficients of the first and second fundamental forms in the basis $(\mathbf{x}_s, \mathbf{x}_t)$.

It is well known (e.g., [do Carmo, 1976, pp. 98]), that $EG - F^2 = |\mathbf{x}_s \times \mathbf{x}_t|^2$. Let us denote the vector $\mathbf{x}_t \times \mathbf{x}_s$ by $\mathbf{n}$. $\mathbf{n}$ is the (non-normalized) normal to the surface. The coefficients e, f, g are given by the following formulas

$$e = \frac{1}{|\mathbf{n}|}\mathbf{n} \cdot \mathbf{x}_{ss} = \frac{1}{|\mathbf{n}|} r(z''r' - r''z')(x'y - y'x),$$

$$f = \frac{1}{|\mathbf{n}|}\mathbf{n} \cdot \mathbf{x}_{st} = 0, \qquad (7.5)$$

$$g = \frac{1}{|\mathbf{n}|}\mathbf{n} \cdot \mathbf{x}_{tt} = \frac{1}{|\mathbf{n}|} z'r^2(x''y' - y''x').$$

So K s given by

$$K = \frac{r^3}{|\mathbf{n}|^4}(z''r' - r''z')(x''y' - y''x')z'(x'y - y'x). \quad (7.6)$$

But the curvature of a planar curve $(u, v) : I \longrightarrow R^2$ is given by

$$\kappa = \frac{u'v'' - v'u''}{(u'^2 + v'^2)^{3/2}}. \qquad (7.7)$$

The proposition therefore follows from the above expression of K. ∎

We are now going to prove the propositions which relate two descriptions of a same surface by different SHGC's (equivalent SHGC's, pivot and slant theorems). We now present the general idea of the proof, which is the same for the three propositions, then give without proof three lemmas, and finally give the detailed proofs of the propositions.

Two SHGC's $(\mathbf{x}_{01}, P_1, (p_1, q_1), S_1, C_1)$ and $(\mathbf{x}_{02}, P_2, (p_2, q_2), S_2, C_2)$ parameterize the same surface iff there exists a bijection $h : (s_1, t_1) \in I_1 \times J_1 \longrightarrow (s_2, t_2) \in I_2 \times J_2$ such that, for all $(s_1, t_1) \in I_1 \times J_1$, we have $\mathbf{x}_1(s_1, t_1) = \mathbf{x}_2 \circ h(s_1, t_1)$, where "o" denotes the operator of composition of two mappings.

As the surface associated to a SHGC is regular, h is a diffeomorphism (i.e., it is differentiable, invertible, and its inverse is differentiable, see [do Carmo, 1976, pp. 70]). To prove our results, we will use this result and the fact that for all $(s_1, t_1) \in I_1 \times J_1$, we have $\mathbf{x}_2 \circ h(s_2, t_2) = \mathbf{x}_1(s_1, t_1)$, and, as the tangent plane of a regular surface is independent of the parameterization, we have $\mathbf{n}_1(s_1, t_1) \times \mathbf{n}_2 \circ h(s_1, t_1) = \mathbf{0}$ (where $\mathbf{n}_i$ denotes the normal to the ith SHGC).

In our proofs, we will often have to prove that s_2 (respectively t_2) is a function of s_1 (respectively t_1) only, which is not a *priori* evident, and in fact would be false for two arbitrary parameterizations of arbitrary regular surfaces. The following two lemmas will be useful for that purpose.

Lemma 7.2: *Let I and J be two open intervals of R and let $f : I \subset R \rightarrow R$ and $g : J \subset R \rightarrow R$ be two real differentiable functions defined on these intervals. Let K be a third open interval of R and $h : I \times K \rightarrow J$ be a differentiable function from $I \times K$ into J such that*

$$\forall (s, t) \in I \times K, g \circ h(s, t) = f(s). \qquad (7.8)$$

Then, if there is no non-empty open interval such that f is constant on that interval, h is a function of s only.

Lemma 7.3: *Let I and J be two open intervals of R and let $(x_1, y_1) : I \subset R \rightarrow R^2$ and $(x_2, y_2) : J \subset R \rightarrow R^2$ be two differentiable simple curves defined on these intervals. Let K be a third open interval of R and $h : I \times K \rightarrow J$ be a differentiable function from $I \times K$ into J such that*

$$\forall (s, t) \in I \times K, \quad x_1(s) y_2 \circ h(s, t) - y_1(s) x_2 \circ h(s, t) = 0. \qquad (7.9)$$

Then, if there is no non-empty open interval such that $x_1'(s) y_1(s) - y_1'(s) x_1(s)$ is zero on that interval, h is a function of s only.

The following lemma will also be useful to characterize the shape of SHGC's parameterizing the same surface.

Lemma 7.4: *Let I and J be two open intervals of R. Let $f_1 : I \subset R \rightarrow R$ and $f_2 : I \subset R \rightarrow R$ be two real functions defined on I, and $g_1 : J \subset R \rightarrow R$ and $g_2 : J \subset R \rightarrow R$ be two real functions defined on I, such that*

$$\forall (s, t) \in I \times J, \quad f_1(s) g_1(t) - f_2(s) g_2(t) = 0. \quad (7.10)$$

Then, if neither f_1 nor g_1 is identically 0, there exists $K \neq 0$, such that for all $s \in I$, $f_1(s) = K f_2(s)$, and for all $t \in J$, $g_2(t) = K g_1(t)$.

We now have all the tools necessary to prove our three propositions.

Proposition 3.1: *The surfaces associated with two SHGC's with the same (non-oriented) cross-section plane and (non-oriented) axis are the same iff the two SHGC's are equivalent.*

Proof of the proposition: Trivially, the (non-oriented) axis and (non-oriented) cross-section plane, and the surface associated with a SHGC are invariant through any of the transformations which relate two equivalent SHGC's. This proves the "if" part of the proposition.

Reciprocally, suppose that the same surface is associated to two SHGC's with the same (non-oriented) axis and cross-section plane. It is clear then that, through one or several of the first four of the equivalence transformations, the orientation of the plane and axis, the origin $\mathbf{x}_0$, and the vectors $(\mathbf{i},\mathbf{j})$ and $\mathbf{a}$ associated with the two SHGC's can be made to coincide.

We have only to prove that in that case, the reference cross-section and the sweeping rule associated with the two SHGC's are obtained through two inverse constant scalings.

Let us consider two SHGC's $(\mathbf{x}_0, P, (p,q), S_1, C_1)$ and $(\mathbf{x}_0, P, (p,q), S_2, C_2)$ with the same origin $\mathbf{x}_0$, plane P and axis direction $\mathbf{a}$. The surfaces associated with these two SHGC's are the same iff there exists a bijection $h : (s_1,t_1) \in I_1 \times J_1 \rightarrow (s_2,t_2) \in I_2 \times J_2$ such that, for any $(s_1,t_1) \in I_1 \times J_1$ and $(s_2,t_2) = h(s_1,t_1)$, we have

$$r_1 \begin{pmatrix} x_1 \\ y_1 \\ 0 \end{pmatrix} - r_2 \begin{pmatrix} x_2 \\ y_2 \\ 0 \end{pmatrix} = (z_2 - z_1) \begin{pmatrix} p \\ q \\ 1 \end{pmatrix}. \qquad (7.11)$$

Using the $\mathbf{k}$ coordinate of the previous equation, we can eliminate $z_2 - z_1 = 0$ from its $\mathbf{i}$ and $\mathbf{j}$ coordinates. Cross-multiplying these coordinates, and using the fact that r_1 and r_2 are strictly positive, we finally get the system of equations

$$x_2 y_1 - y_2 x_1 = 0, \qquad (7.12)$$

$$z_2 = z_1. \qquad (7.13)$$

As remarked earlier, h is a diffeomorphism, and is in particular differentiable. Clearly, the two above equations satisfy the conditions of lemmas (7.2) and (7.3), and we conclude that s_2 is a *function* of s_1 *only* and that t_2 is a function of t_1 only.

We can now suppose without loss of generality that both the SHGC's are parameterized by the same parameters (s_1,t_1). From the above equations, we have in particular that, for all $(s_1,t_1) \in I_1 \times J_1$, $r_1 x_1 = r_2 x_2$ and $r_1 y_1 = r_2 y_2$

This time, the hypotheses of lemma (7.4) are clearly satisfied, and we conclude that there exists a non-zero constant K such that $r_1 = K r_2$, $x_2 = K x_1$, and $y_2 = K y_1$. This completes the proof of the proposition. ■

Proposition 5.1: *The surface associated with a non linear SHGC cannot be described by another SHGC with a different axis, and the same cross-section plane.*

Proof of the proposition: Let us consider two SHGC's $(\mathbf{x}_{01}, P, (p_1,q_1), S_1, C_1)$ and $(\mathbf{x}_{02}, P, (p_2,q_2), S_2, C_2)$ with the same cross-section plane P.

$$\forall (s_1,t_1) \in I_1 \times J_1, \quad 0 = \mathbf{x}_2 - \mathbf{x}_1 = \begin{pmatrix} x_{02} - x_{01} \\ y_{02} - y_{01} \\ z_{02} - z_{01} \end{pmatrix}$$

$$+ \begin{pmatrix} r_2 x_2 - r_1 x_1 \\ r_2 y_2 - r_1 y_1 \\ 0 \end{pmatrix} + \begin{pmatrix} z_2 p_2 - z_1 p_1 \\ z_2 q_2 - z_1 q_1 \\ z_2 - z_1 \end{pmatrix}. \qquad (7.14)$$

We can suppose without loss of generality that $\mathbf{x}_{01}$ and $\mathbf{x}_{02}$ have the same $\mathbf{k}$ coordinate z_0 (by translating if necessary one of the origins along the corresponding axis). In particular, this implies that for all $(s_1,t_1) \in I_1 \times J_1$, we have $z_1(s_1) = z_2(s_2)$.

We have seen earlier that this implies that s_2 is a function of s_1 only, and we can from now on assume without loss of generality that r_2 and z_2 are parameterized by s_1.

Similarly, we now prove that t_2 is a function of t_1 only. The $\mathbf{k}$ component n_z of $\mathbf{n}_1 \times \mathbf{n}_2$ is

$$n_z = -(y_2'(t_2)x_1'(t_1) - x_2'(t_2)y_1'(t_1))(r_1 r_2 z_1' z_2') =$$

$$= -(y_2'(t_2)x_1'(t_1) - x_2'(t_2)y_1'(t_1))(r_1 r_2 z_1'^2). \qquad (7.15)$$

As by hypothesis, z_1' is not identically 0, we can restrict ourselves to an interval I where z_1' is nowhere 0. As r_1 and r_2 are never zero, we can use on that interval, lemma (7.3) (its conditions are once again clearly satisfied). It follows that t_2 is a function of t_1 only.

To complete the proof, we are going to use again the fact that $\mathbf{x}_2 - \mathbf{x}_1$ is a constant (zero) vector. As t_2 is a function of t_1 only, and s_2 is a function of s_1 only, we can suppose without loss of generality that both SHGC's are parameterized by (s_1,t_1). In the coordinate system P, we have

$$\mathbf{0} = \mathbf{x}_2 - \mathbf{x}_1 = \begin{pmatrix} x_{02} - x_{01} \\ y_{02} - y_{01} \\ 0 \end{pmatrix} + \begin{pmatrix} r_2 x_2 - r_1 x_1 \\ r_2 y_2 - r_1 y_1 \\ 0 \end{pmatrix}$$

$$+ z \begin{pmatrix} p_2 - p_1 \\ q_2 - q_1 \\ 0 \end{pmatrix} = \begin{pmatrix} X \\ Y \\ 0 \end{pmatrix} (s,t). \qquad (7.16)$$

In particular, we have $0 = X_t = x_2' r_2 - x_1' r_1$ and $0 = Y_t = y_2' r_2 - y_1' r_1$. We deduce as before that

$r_1 = Kr_2, x'_2 = Kx'_1, y'_2 = Ky'_1$ for some (non-zero) constant K.

Substituting in X and Y, we get that $X = (x_{02}-x_{01})+ ar_2 + z(p_2 - p_1)$ and $Y = (y_{02} - y_{01}) + br_2 + z(q_2 - q_1)$ for some constants a and b. As both X and Y must be 0, and by hypothesis (z, r_2) is not linear, it follows that $a = b = 0$, $p_1 = p_2$, $q_1 = q_2$, $x_{01} = x_{02}$, and $y_{01} = y_{02}$. This completes the proof of the proposition. ■

Proposition 5.2: *The surface associated with a non linear SHGC cannot be described by another SHGC with the same axis, and a different cross-section plane.*

Proof of the proposition: Let us consider two SHGC's with the same axis. We can suppose without loss of generality that their origins coincide (by translating if necessary the origin of one of them along the axis). We can also suppose without loss of generality that the vectors **i** of the associated coordinate systems P_1 and P_2 are the same (by rotating if necessary the two coordinate systems around their **k** vectors).

The two SHGC's are $(\mathbf{x}_0, P_1 = (\mathbf{i}, \mathbf{j}_1, \mathbf{k}_1), \mathbf{a}, S_1, C_1)$ and $(\mathbf{x}_0, P_2 = (\mathbf{i}, \mathbf{j}_2, \mathbf{k}_2), \mathbf{a}, S_2, C_2)$. Notice that **a** does not have the same coordinates in the planes P_1 and P_2. Let (p, q) be its coordinates in P_1. Let α be the angle of the rotation around **i** which maps P_1 onto P_2, we have

$$\mathbf{j}_2 = \cos\alpha\mathbf{j}_1 + \sin\alpha\mathbf{k}_1; \mathbf{k}_2 = -\sin\alpha\mathbf{j}_1 + \cos\alpha\mathbf{k}_1. \quad (7.17)$$

In particular, the coordinates of a point $\mathbf{x}_2(s_2, t_2)$ of the second SHGC in the coordinate system $(\mathbf{x}_0, \mathbf{i}, \mathbf{j}_1, \mathbf{k}_1)$ are

$$\mathbf{x}_2(s_2, t_2) = r_2 \begin{pmatrix} x_2 \\ y_2\cos\alpha \\ y_2\sin\alpha \end{pmatrix} + z_2 \begin{pmatrix} p \\ q \\ 1 \end{pmatrix}. \quad (7.18)$$

The coordinates of $\mathbf{n}_2$ in P_1 are given by

$$\mathbf{n}_2(s_2, t_2) = r_2 r'_2(x'_2 y_2 - y'_2 x_2) \begin{pmatrix} 0 \\ -\sin\alpha \\ \cos\alpha \end{pmatrix}$$

$$+ r_2 z'_2 \begin{pmatrix} y'_2(\cos\alpha - q\sin\alpha) \\ -x'_2 + y'_2 p\sin\alpha \\ x'_2 q - y'_2 p\cos\alpha \end{pmatrix}. \quad (7.19)$$

Writing $\mathbf{x}_2 = \mathbf{x}_1$, we get

$$r_1 \begin{pmatrix} x_1 \\ y_1 \\ 0 \end{pmatrix} - r_2 \begin{pmatrix} x_2 \\ y_2\cos\alpha \\ y_2\sin\alpha \end{pmatrix} = (z_2 - z_1) \begin{pmatrix} p \\ q \\ 1 \end{pmatrix} \quad (7.20)$$

Eliminating $z_2 - z_1$ by using the $\mathbf{k}_1$ coordinate of this equation, and cross-multiplying its **i** and $\mathbf{j}_1$ coordinates, we get (as r_1 and r_2 are strictly positive)

$$\forall (s_1, t_1) \in I_1 \times J_1, \quad x_1(t_1)y_2(t_2)(\cos\alpha - q\sin\alpha)$$

$$-y_1(t_1)(x_2(t_2) - y_2(t_2)p\sin\alpha) = 0. \quad (7.21).$$

Let $Y_2 = y_2(\cos\alpha - q\sin\alpha)$ and $X_2 = x_2 - y_2 p\sin\alpha$, we have $X'_2 Y_2 - Y'_2 X_2 = (\cos\alpha - q\sin\alpha)(x'_2 y_2 - y'_2 x_2)$. The quantity $(\cos\alpha - q\sin\alpha)$ is equal to $\mathbf{a} \cdot \mathbf{k}_2$, and is therefore different from zero (the axis of a SHGC is not in the cross section plane of this SHGC).

The hypotheses of lemma (7.3) are therefore satisfied, and we conclude that t_1 is a function of t_2 only. We now write that $\mathbf{n}_1 \times \mathbf{n}_2 = \mathbf{0}$. In particular, writing that the $\mathbf{k}_1$ component of this cross product is zero, we obtain that

$$z'_1(r'_2(y'_1\sin\alpha(x'_2 y_2 - x_2 y'_2)) - z'_2(x'_1 y'_2(\cos\alpha - q\sin\alpha)$$

$$-y'_1(x'_2 - y'_2 p\sin\alpha))) = 0. \quad (7.22)$$

Let (s_0, t_0) be a point where $z'_1 \circ s_1(s_0, t_0) \neq 0$, (such a point exists as z'_1 is not identically 0, and h is a bijection), and $I_0 \times J_0$ an open rectangle centered at (s_0, t_0) such that for all $(s_2, t_2) \in I_0 \times J_0$, $z'_1 \circ s_1(s_2, t_2) \neq 0$ (this rectangle exists because $z'_1 \circ s_1$ is continuous. We then have

$$\forall (s_2, t_2) \in I_0 \times J_0, \qquad r'_2(y'_1\sin\alpha(x'_2 y_2 - x_2 y'_2))$$

$$-z'_2(x'_1 y'_2(\cos\alpha - q\sin\alpha) - y'_1(x'_2 - y'_2 p\sin\alpha)) = 0. \quad (7.23)$$

Let us suppose that $\sin\alpha \neq 0$. The hypotheses of lemma (7.4) are then trivially satisfied with $f_1 = r'_2$, $g_1 = y'_1\sin\alpha(x'_2 y_2 - x_2 y'_2)$, $f_2 = z'_2$, and $g_2 = x'_1 y'_2(\cos\alpha - q\sin\alpha) - y'_1(x'_2 - y'_2 p\sin\alpha)$.

In particular this implies that (r_2, z_2) is linear on I_0, which contradicts the hypothesis. It follows that $\sin\alpha = 0$, which completes the proof of the proposition. ■

Ribbons, Symmetries, and Skewed Symmetries*

Jean Ponce

Robotics Laboratory
Department of Computer Science
Stanford University
Stanford, California 94305

Abstract

Following Rosenfeld [Rosenfeld 1986], we compare in this short paper Blum, Brooks, and Brady ribbons. We prove that Blum and Brady ribbons are not, in general, Brooks ribbons. Conversely, we prove that Brooks ribbons are, in general, neither Blum nor Brady ribbons. For Blum and Brooks ribbons, it is in principle trivial to decide whether two contour points may form a ribbon pair: they have to form a local symmetry, i.e., the angles between the line joining these two points and the outward normals to the contour at these points must be the same. This property is not true for Brooks ribbons. Is it then possible to characterize locally the pairs of contour points which form a Brooks ribbon pair? Using the curvature of the contour of a Brooks ribbon, we prove that the answer to this question is yes for some classes of Brooks ribbons, including skewed symmetries. This result has potential practical applications for three-dimensional shape recovery from image contours [Kanade, 1981].

Introduction

Ribbons are planar shapes generated by sweeping a geometric figure along a plane curve. Rosenfeld [Rosenfeld, 1986], has compared three popular classes of ribbons: Blum [Blum, 1967], Brooks [Brooks, 1981], and Brady [Brady and Asada, 1984] ribbons.

Rosenfeld has studied the properties of these ribbons from the standpoints of both generation and recovery, and compared their relative advantages and disadvantages. He has also proved that there are Brady ribbons that are not Brooks or Blum ribbons and that, for straight spines, ignoring the ends, every Blum ribbon is a Brooks ribbon, and every Brooks ribbon is a Brady ribbon.

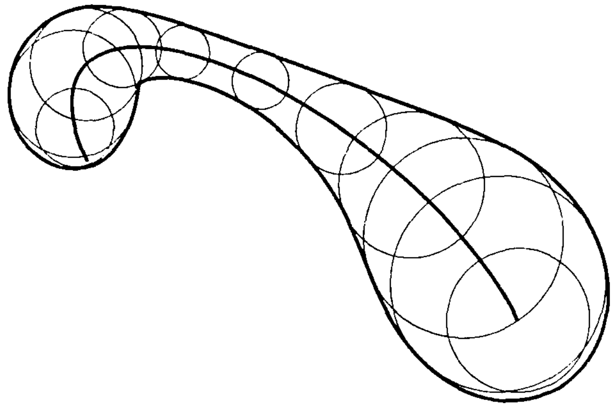


Figure 1. A Blum ribbon, generated by sweeping a disc along a plane curve. In this figure, the boundary of the ribbon has been computed by using the expression for the envelope given in section 2.

In this paper, we consider Blum, Brooks, and Brady ribbons with curved spines. We prove that, in this case, Blum ribbons are not, in general, Brooks ribbons, and that Brooks ribbons are not, in general, Brady ribbons. As Blum ribbons are always Brady ribbons, it follows immediatly that Brady ribbons are not, in general, Brooks ribbons, and that Brooks ribbons are not, in general, Brady ribbons.

In particular, this implies that we cannot use segmentation algorithms developed for finding Blum and Brady ribbons (e.g., [Bookstein, 1979], [Serra, 1982], [Brady and Asada, 1984]) to segment images containing Brooks ribbons. However, certain classes of Brooks ribbons, e.g., skewed symmetries, are of practical importance, as they can be used for three-dimensional shape recovery from image contours [Kanade, 1981].

A particular class of algorithms [Brady and Asada, 1984] for finding Blum and Brady ribbons in images is based on the fact that two points on the boundary of such a ribbon that correspond to the same axis point form a local symmetry. Such ribbon pairs can be found

*Support for this work was provided in part by the Air Force Office of Scientific Research under contract F33615-85-C-5106 and by the Advanced Research Projects Agency of the Department of Defense under Knowledge Based Vision ADS subcontract S1093 S-1.

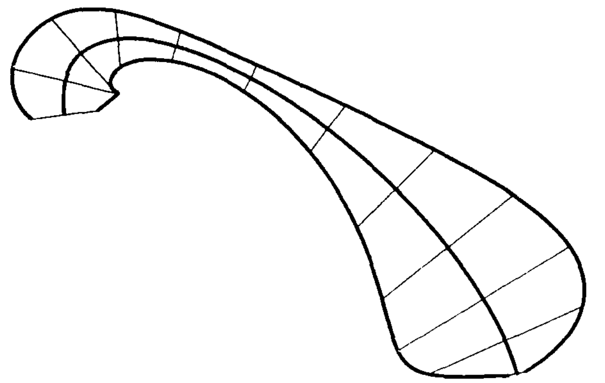


Figure 2. A Brooks ribbon, generated by sweeping a line segment along a plane curve. The angle between this line segment and the tangent to the curve is constant.

by testing all possible pairs of contour points for local symmetry, and these ribbons pairs can in turn be grouped into ribbons.

If we can find such a "local signature" (i.e., characterize ribbon pairs based on image features such as angles, distances, and curvatures, which are measurable locally in images) for Brooks ribbons, then we can use similar algorithms to segment images containing Brooks ribbons.

Is there any such form of local signature for Brooks ribbons? Using contour curvature, we show that the answer is no in the most general case, but yes for some classes of Brooks ribbons. Skewed symmetries are one of these classes.

The paper is organized as follows. In section 1, we define more precisely Blum, Brooks, and Brady ribbons. We then prove (section 2) that Blum ribbons are not Brooks ribbons, and (section 3) that Brooks ribbons are not Brady ribbons. In section 4, we calculate the curvature of the contour of a Brooks ribbon, and show that it can be used to characterize ribbon pairs for some classes of Brooks ribbons. Finally, in section 5, we use these results to give a local characterization of pairs of points which form skewed symmetries, and sketch an algorithm for finding skewed symmetries in an image.

1. Ribbons

Following Rosenfeld [Rosenfeld, 1986], we call ribbon a plane shape generated by sweeping a geometric figure along a spine. The size and the orientation of this figure may change as it moves.

A ribbon pair is a pair of points on the boundary of a ribbon which correspond to the same axis point.

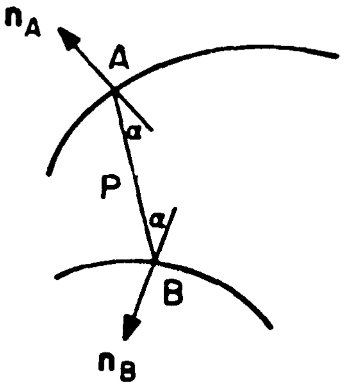


Figure 3. Brady ribbons are defined by local symmetries.

In this paper, we will characterize ribbons in terms of ribbon pairs. Notice that a given point may participate in several ribbon pairs.

Blum ribbons ([Blum, 1967],[Blum and Nagel, 1978], see Figure 1) are generated by a disk centered on the spine. Correspondingly, Blum ribbon pairs are made of points such that there exists a disc included in the ribbon and tangent to its boundary at these two points.

Brooks ribbons ([Brooks, 1981], see Figure 2) are generated by line segments centered on the spine such that the angle between the line segments and the tangent to the spine is constant. Brooks ribbon pairs correspond to the end points of these segments.

Brooks ribbons are two-dimensional versions of generalized cylinders [Binford, 1971]. Notice that Rosenfeld [Rosenfeld, 1986] considers in his analysis the class of Brooks ribbons for which the angle between the generators and the spine is a right angle.

Brady ribbons ([Brady, 1984], see Figure 3) are generated by line segments which make equal angles with the sides of the ribbon. A Brady ribbon pair consists of two points such that the angles between their outward normal and the line joining the points are the same, i.e. these points form a local symmetry.

It is not a priori evident that Brady ribbons have a smooth spine. Giblin and Brassett have, however, proved that it is in general true [Giblin and Brassett, 1985]. Conversely, Brooks ribbons have by definition a smooth spine, but it is not clear what the ribbon associated to a given arbitrary shape is.

In the sequel, we will suppose that the spine of a ribbon is a smooth plane curve $\mathbf{x}(s)$ parameterized by its arc length s. The corresponding unit tangent, normal, and curvature will be denoted by $\mathbf{t}$, $\mathbf{n}$, κ.

We will use the fact that, from elementary differential geometry [do Carmo, 1976], we have

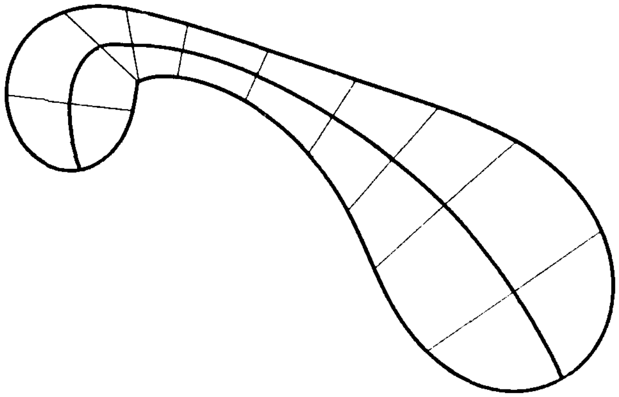


Figure 4. This Blum ribbon is not a Brooks ribbon. The axis drawn in this figure is the Brady/Brooks axis corresponding to the envelope of the Blum ribbon. The angle between the line generators and the axis is not constant.

$$\frac{d\mathbf{x}}{ds} = \mathbf{t}, \frac{d\mathbf{t}}{ds} = \kappa\mathbf{n}, \frac{d\mathbf{n}}{ds} = -\kappa\mathbf{t}.$$

2. Blum ribbons are not Brooks ribbons

We now prove that Blum ribbons are not, in general Brooks ribbons. An immediate corollary of this result is that Brady ribbons (at least those which may also be described as Blum ribbons) are not, in general, Brooks ribbons.

To prove this result, we consider Blum ribbons whose spine is a simple smooth curve. The boundary of a Blum ribbon is given by its envelope. We derive an expression for the envelope, and show that the angle between the line joining two points on this envelope which form a Blum ribbon pair and the tangent to the spine is not constant (Figure 4). This proves that the ribbon is not a Brooks ribbon.

Suppose the Blum spine given by a plane curve $\mathbf{x}(s)$ parameterized by its arc length s, and let $\mathbf{t}$, $\mathbf{n}$, and κ denote the corresponding tangent, normal, and curvature. Consider now a function $r(s)$ and the region $\mathbf{r}(s, \theta)$ defined by

$$\mathbf{r}(s, \theta) = \mathbf{x}(s) + r(s)(\cos\theta\mathbf{t} + \sin\theta\mathbf{n}).$$

This region is a Blum ribbon, and $\mathbf{x}(s)$ is its spine. The boundary of this ribbon is given by its envelope $\mathbf{e}(s)$, which is tangent to the curves $\mathbf{r}(s, \theta)$ for every s (see [Faux and Pratt, 1979]). Methods for characterizing the envelope can be found in the literature. In particular, we must have

$$\frac{\partial\mathbf{r}}{\partial\theta} \times \frac{d\mathbf{e}}{ds} = 0,$$

where "×" denotes the operator which associates to two vectors their determinant. Substituting

$$\frac{d\mathbf{e}}{ds} = \frac{d[\mathbf{r}(s, \theta(s))]}{ds} = \frac{d\theta}{ds}\frac{\partial\mathbf{r}}{\partial\theta} + \frac{\partial\mathbf{r}}{ds},$$

in the previous equation, we get

$$\frac{d\mathbf{r}}{ds} \times \frac{d\mathbf{r}}{d\theta} = 0.$$

We have

$$\frac{d\mathbf{r}}{ds} = (1 + r'\cos\theta - \kappa r\sin\theta)\mathbf{t} + (r'\sin\theta + \kappa r\cos\theta)\mathbf{n},$$

$$\frac{d\mathbf{r}}{d\theta} = r(-\sin\theta\mathbf{t} + \cos\theta\mathbf{n}).$$

It follows that the envelope is given by the points (s, θ) which verify

$$\cos\theta + r'(s) = 0.$$

This equation has solutions iff $|r'(s)| \leq 1$, and two solutions in that case. The corresponding envelope points are then $\mathbf{x}_\epsilon(s), \epsilon = \mp 1$, with

$$\mathbf{x}_\epsilon(s) = \mathbf{x} + r(-r'\mathbf{t} + \epsilon\sqrt{1 - r'^2}\mathbf{n}).$$

The pairs $(\mathbf{x}_{-1}, \mathbf{x}_{+1})$ define a Blum ribbon pair, and therefore a Brady ribbon pair. We are now going to compute the axis of the associated Brady ribbon, and then prove that the pair is not a Brooks ribbon pair. The axis of the Brady ribbon is given by

$$\mathbf{x}_0 = \frac{1}{2}(\mathbf{x}_{-1} + \mathbf{x}_{+1}) = \mathbf{x} - rr'\mathbf{t}.$$

Note that, as expected, the Blum and Brady axes don't coincide, unless the Blum axis is straight or the function r is constant. Let us now show that this ribbon pair is not a Brooks ribbon pair. If it was one, its Brooks axis and Brady axis would coincide (each of them is the locus of the mid-points of the ribbon pair).

But the angle α between the tangent to the axis of a Brooks ribbon and the line joining the points of any pair of this ribbon is constant. Let us calculate this angle. We first calculate the tangent to $\mathbf{x}_0(s)$. We have

$$\frac{d\mathbf{x}_0}{ds} = (1 - r'^2 - rr'')\mathbf{t} - \kappa rr'\mathbf{n}.$$

So the unit tangent $\mathbf{t}_0$ to the Brady axis is

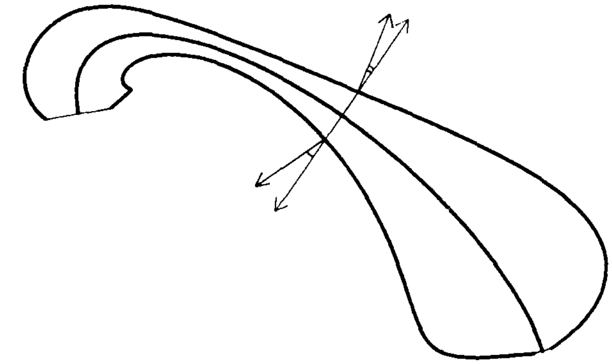


Figure 5. This Brooks ribbon is not a Brady ribbon: the angles between a generator and its sides are not the same.

$$\mathbf{t}_0 = \frac{1}{\sqrt{(1-r'^2-rr'')^2+(\kappa r r')^2}}((1-r'^2-rr'')\mathbf{t}-\kappa r r'\mathbf{n}).$$

The direction of the line joining $\mathbf{x}_{-1}$ to $\mathbf{x}_{+1}$ is $\mathbf{n}$. We have therefore

$$\cos\alpha = \frac{-\kappa r r'}{\sqrt{(1-r'^2-rr'')^2+(\kappa r r')^2}}.$$

This angle is clearly non-constant in general. A sufficient condition for it to be constant (and in that case the angle is $\pi/2$) is that $\kappa = 0$ or $r' = 0$, as could be expected. We have just proved that Blum ribbons and Brady ribbons are, in general, not Brooks ribbons.

3. Brooks ribbons are not Brady ribbons

We now prove that Brooks ribbons are not, in general, Brady ribbons. An immediate corollary of this result is that Brooks ribbons are not, in general, Blum ribbons.

To prove this result, we consider a Brooks ribbon, and derive an expression for points on its boundary which form a ribbon pair. We prove that the angles between the outward normals to the contour at these points and the line joining the points are not the same (Figure 5). The result follows immediatly.

Consider an angle θ, a function $r(s)$ and the curve $\mathbf{x}_\epsilon(s), \epsilon = \mp 1$, defined by

$$\mathbf{x}_\epsilon(s) = \mathbf{x}(s) + \epsilon r(s)(\cos\theta\,\mathbf{t}(s) + \sin\theta\,\mathbf{n}(s)).$$

We have just defined a Brooks ribbon of spine $\mathbf{x}(s)$ whose sides are given by $\mathbf{x}_{-1}(s)$ and $\mathbf{x}_{+1}(s)$. For a given s, the pair $(\mathbf{x}_{-1}, \mathbf{x}_{+1})$ is a ribbon pair.

Let s_ϵ, and $\mathbf{t}_\epsilon$ denote the associated arc length and unit tangent. We have

$$\frac{d\mathbf{x}_\epsilon}{ds} = [1+\epsilon(r'\cos\theta-\kappa r\sin\theta)]\mathbf{t}+\epsilon(r'\sin\theta+\kappa r\cos\theta)\mathbf{n}.$$

Therefore,

$$\frac{ds_\epsilon}{ds} = |\frac{d\mathbf{x}_\epsilon}{ds}| = \sqrt{1+2\epsilon(r'\cos\theta-\kappa r\sin\theta)+\kappa^2r^2+r'^2},$$

and

$$\mathbf{t}_\epsilon = \frac{d\mathbf{x}_\epsilon}{ds_\epsilon} = \frac{ds}{ds_\epsilon}\frac{d\mathbf{x}_\epsilon}{ds}.$$

The direction of the line joining $\mathbf{x}_{-1}$ and $\mathbf{x}_{+1}$ is $\mathbf{c} = \cos\theta\mathbf{t} + \sin\theta\mathbf{n}$. Let α_ϵ be the angle between $\mathbf{c}$ and $\mathbf{t}_\epsilon$, we deduce from the above expression that

$$\cos\alpha_\epsilon = \frac{\cos\theta+\epsilon r'}{\sqrt{1+2\epsilon(r'\cos\theta-\kappa r\sin\theta)+\kappa^2r^2+r'^2}},$$

$$\sin\alpha_\epsilon = \frac{\sin\theta-\epsilon\kappa r}{\sqrt{1+2\epsilon(r'\cos\theta-\kappa r\sin\theta)+\kappa^2r^2+r'^2}}.$$

The points $\mathbf{x}_{-1}$ and $\mathbf{x}_{+1}$ form a Brady ribbon pair (i.e., a local symmetry) iff $\alpha_{-1} = \pi - \alpha_{+1}$, or equivalently $\tan\alpha_{-1} = -\tan\alpha_{+1}$. This condition can be easily rewritten

$$\kappa r r' = -\sin\theta\cos\theta.$$

It follows that Brooks ribbons are not, in general, Brady ribbons. In the case studied by Rosenfeld [Rosenfeld, 1986], we have $\theta = \pi/2$ (right ribbon), and a Brooks ribbon is a Brady ribbon iff $\kappa = 0$ (straight axis) or $r' = 0$ (constant cross-section). In the general case, even Brooks ribbons with a straight axis or a constant cross-section are not Brady ribbons.

4. Local signature of Brooks ribbons

We have shown that Brooks ribbons are not Brady ribbons. In particular they don't form local symmetries, and algorithms like those used to find smooth local symmetries [Brady and Asada, 1984] cannot be used to segment images containing Brooks ribbons.

Is it possible to find another "local signature" characterizing Brooks ribbon pairs based on quantities measurable locally in images (i.e., distances, angles, curvatures)? To answer this question, we now calculate the curvature κ_ϵ of a Brooks ribbon.

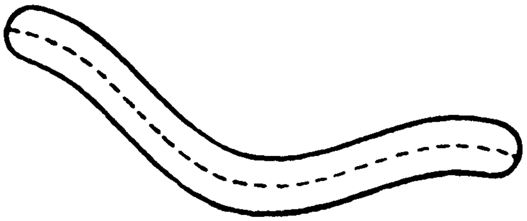


Figure 6. A worm, i.e., a Brooks ribbon with a constant cross-section.

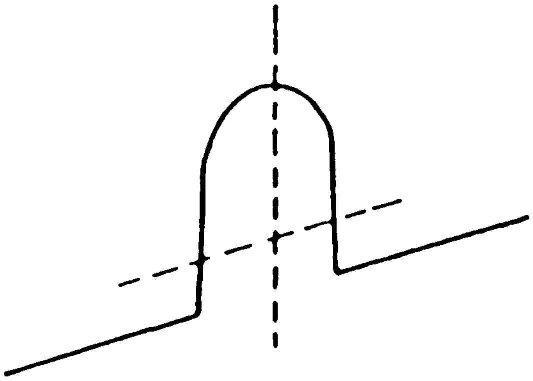


Figure 7. A skewed symmetry, i.e., a Brooks ribbon with a straight spine.

It is well known that the curvature of a plane curve $\mathbf{y}(t)$ (not necessarily parameterized by its arc length) is given by

$$\kappa(t) = \frac{\mathbf{y}' \times \mathbf{y}''}{|\mathbf{y}'|^3},$$

where "$\times$" denotes the operator which associates to two vectors their determinant (see, for example [do Carmo, 1976], pp. 25).

We have

$$\frac{d^2\mathbf{x}_\epsilon}{ds^2} = \epsilon((r'' - \kappa^2 r)\cos\theta - (\kappa' r + 2\kappa r')\sin\theta)\mathbf{t}$$
$$+[\kappa + \epsilon((r'' - \kappa^2 r)\sin\theta + (\kappa' r + 2\kappa r')\cos\theta)\mathbf{n}.$$

It follows that

$$\kappa_\epsilon |\mathbf{x}'_\epsilon|^3 = \kappa(1 + 2\epsilon(r'\cos\theta - \kappa r\sin\theta) + \kappa^2 r^2 + r'^2)$$
$$+(\epsilon\cos\theta + r')(\kappa' r + \kappa r') + (\epsilon\sin\theta - \kappa r)r''.$$

As could be expected, the curvature κ_ϵ depends on κ,κ',r,r', r'', and θ. On the other hand, we can measure in the images the quantities α_ϵ, κ_ϵ ($\epsilon = \mp 1$), and r.

We have therefore six unknowns for five measures, so it is not possible, in general, to characterize Brooks ribbon pairs based only on local information. If some of the Brooks ribbon parameters are fixed, however, this characterization becomes possible.

We now consider such an example, the case of a worm (Figure 6), i.e., a Brooks ribbon with a constant cross-section ($r' = r'' = 0$). We now have four unknowns and five measures, so we can characterize ribbon pairs. Th^e expressions of $\cos\alpha_\epsilon$, $\tan\alpha_\epsilon$, and κ_ϵ simplify into

$$\cos\alpha_\epsilon = \frac{\cos\theta}{\sqrt{1 - 2\epsilon\kappa r\sin\theta + \kappa^2 r^2}},$$

$$\tan\alpha_\epsilon = \frac{\sin\theta - \epsilon\kappa r}{\cos\theta},$$

$$\kappa_\epsilon = \frac{\kappa(1 - 2\epsilon\kappa r\sin\theta + \kappa^2 r^2) + \epsilon\kappa' r\cos\theta}{(1 - 2\epsilon\kappa r\sin\theta + \kappa^2 r^2)^{3/2}}.$$

It follows that

$$\frac{\kappa_{+1} + \frac{\cos\alpha_{+1}}{2r}(\tan\alpha_{+1} - \tan\alpha_{-1})}{\kappa_{-1} + \frac{\cos\alpha_{-1}}{2r}(\tan\alpha_{+1} - \tan\alpha_{-1})} = -\left[\frac{\cos\alpha_{+1}}{\cos\alpha_{-1}}\right]^3.$$

The above expression is a necessary local condition that a pair of points must verify to form a worm ribbon pair.

5. Skewed symmetries

We finally consider a simple (but important) class of Brooks ribbons: skewed symmetries (Figure 7). It has been shown [Kanade, 1981], that skewed symmetries can be used to recover the three-dimensional shape of objects with planar faces and bilateral symmetries from their image contours.

Unfortunately, it is difficult to find skewed symmetries in an image (however, see [Friedberg, 1986], for a method based on moments). In this section, we prove that skewed symmetries have a local signature, and propose an algorithm for finding them.

A skewed symmetry is a Brooks ribbon with a straight axis. In that case, $\kappa = 0$ and $\kappa' = 0$. So, again, we have four unknowns for five measures. In particular, we have

$$\sin\alpha_\epsilon = \frac{\sin\theta}{\sqrt{1 + 2\epsilon r'\cos\theta + r'^2}},$$

and

$$\kappa_\epsilon = \frac{\epsilon r'' \sin\theta}{(1 + 2\epsilon r' \cos\theta + r'^2)^{3/2}}.$$

It follows immediatly that

$$\frac{\kappa_{+1}}{\kappa_{-1}} = -\left[\frac{\sin\alpha_{+1}}{\sin\alpha_{-1}}\right]^3.$$

This property is a necessary condition, based on local image measures, that two points must verify to form a skewed symmetry. It can therefore be used to find skewed symmetries in an image. A simple algorithm would be to test every possible pair of contour points, check whether they verify the above condition, and group the ribbon pairs into ribbons. This is analogous to the $O(n^2)$ algorithm for finding smooth local symmetries (see [Brady and Asada, 1984]).

Instead, we propose to use the method of projections [Nevatia and Binford, 1977]. This method can be summarized as follows: discretize the possible orientations of local ribbon axes; for each of these directions, project all contour points into buckets, and verify the above condition for points within the same bucket only; group the resulting ribbon pairs into ribbons.

This method has been implemented for finding Brady ribbons in [Sumanaweera et al., 1988]. Its advantages are its complexity ($O(k.n)$, where k is the number of discretized orientations and n the number of edge points), and an easy grouping of ribbon pairs into ribbons (the neighborood information is preserved during projection). We are currently working on extending this method to skewed symmetries.

References

1. Binford, T.O, "Visual perception by computer", in IEEE Conference on Systems and Cybernetics, Miami, Fla., December 1971.

2. Blum, H., "A transformation for extracting new descriptors of shape", in **Models for the perception of speech and visual form**, W. Wathen-Dunn (ed.), pp. 362-380, MIT Press, Cambridge, Mass., 1967.

3. Blum, H., and Nagel, R.N., "Shape description using weighted symmetric axis features", Pattern Recognition 10, pp. 167-180, 1978.

4. Bookstein, F., "The line skeleton", Computer Graphics and Image Processing 11, pp. 123-137, 1979.

5. Brady, J.M., and Asada, H., "Smoothed local symmetries and their implementation", International Journal of Robotics Research 3, No. 3, 1984.

6. Brooks, R.A., "Symbolic reasoning among 3-D models and 2-D images", Artificial Intelligence 17, pp. 285-348, 1981.

7. do Carmo, M.P, "**Differential geometry of curves and surfaces**", Prentice-Hall, 1976.

8. Faux, I.D., and Pratt, M.J., "Computational geometry for design and manufacture", R. Ellis Horwood, 1979.

9. Friedberg, S.A,"Finding axes of skewed symmetry", Computer Vision, Graphics, and Image Processing 34, pp. 138-155, 1986.

10. Giblin, P.J., and Brassett, S.A., "Local symmetry of plane curves", American Mathematical Monthly 92, pp. 689-707, 1985.

11. Kanade, T., "Recovery of the 3-D shape of an object from a single view", Artificial Intelligence 17, pp. 409-460, 1981. 1986.

12. Rosenfeld, A., "Axial representatons of shape", Computer Vision, Graphics, and Image Processing 33, pp. 156-173, 1986.

13. Serra, J., "**Image analysis and mathematical morphology**", Academic Press, 1982.

14. Sumamaweera, T., Healey, G., Lee, B., and Ponce, J., "Integrated segmentation using geometrical and physical constraints", Proc. IU Workshop, 1988.

SYMBOLIC PIXEL LABELING FOR CURVILINEAR FEATURE DETECTION

John Canning
J. John Kim
Nathan Netanyahu
Azriel Rosenfeld

Computer Vision Laboratory
Center for Automation Research
University of Maryland
College Park, MD 20742-3411

ABSTRACT

This paper describes a method of detecting thin curvilinear features in an image based on a detailed analysis of the local gray level patterns at each pixel. This allows operations such as thinning and gap filling to be based on more accurate information.

Many types of images contain thin curvilinear features; for example, aerial photographs contain numerous features such as roads and streams. This paper describes a method of detecting thin curvilinear features in an image based on a detailed analysis of the local gray level patterns at each pixel. As we shall see, this allows operations such as thinning and gap filling to be based on more accurate information.

The traditional approach [1] to extracting thin curvilinear features from an image is usually along the following lines:

a) Apply local line detectors to the image.

b) Link the detected line pixels into connected components.

c) Thin the components using standard binary-image thinning techniques.

d) Fill gaps in the components using "good continuation" interpolation.

A serious weakness of this approach is that at step (a) it decides, at each pixel, whether a line passes through it, and if so, in what direction, based on the absolute and relative strengths of a set of line detector responses. This decision is made independently for each pixel. Thereafter, in steps (b-d), the thin region is treated as a set of pixels; the local evidence which led to the selection of those pixels is discarded. [A somewhat more flexible approach estimates probabilities of lines in various directions at each pixel, and iteratively adjusts these estimates based on the estimates at neighboring pixels. This "relaxation" approach allows the neighbors of the pixel to influence the decision, but it loses information by combining numerical measures of confidence, and in any case it still does make a final decision and then treats the thin region as a set of pixels.]

Our approach tries to avoid making decisions at the individual pixels, even with the help of their neighbors. Instead, it examines the neighborhood of each pixel and identifies *all* local patterns of gray levels in that neighborhood that are consistent with the presence of a line. (Since the line may be more than one pixel thick, these include all the patterns that are consistent with the presence of an edge.) These patterns are found by examining all possible thresholdings of the 3×3 neighborhood and selecting those of the resulting binary patterns ("masks") that are line- or edge-like. These patterns are shown in Figure 1.

Each pixel now has associated with it a set (possibly empty) of line- or edge-like masks. Note that there can be more than one of these at a given pixel, since two different thresholds can both yield edge- (or line-)like binary patterns, as we see if we apply the threshclds 1 and 3 to the 3×3 neighborhood

$$\begin{array}{ccc}2&2&4\\2&2&4\\0&0&2\end{array}$$

However, there cannot be very many of them; if the gray levels in the neighborhood are $z_1, \ldots, z_9$, where $0 = z_0 \leq z_1 \leq z_2 \leq \cdots \leq z_9 \leq z_{10} = 255$, the patterns produced by any threshold between z_i and z_{i+1} are all the same, and some of these (e.g., for $i = 0$ or 9) are not line- or edge-like.

Now that we have associated masks with the pixels, we can use the mask information in performing steps analogous to (b-d) rather than simply working with sets of pixels. As we shall now see, this leads to improved results.

The support of the Defense Mapping Agency under Contract DMA-85-C-0007 is gratefully acknowledged, as is the help of Sandy German in preparing this paper

We begin by examining the masks at pairs of adjacent pixels, and checking pairs of these masks for compatibility with the presence of a line (or edge). Figure 2 shows sets of pairs that are compatible with the presence of a thin line (the "extend" and "maybe-extend" pairs) or with a line two pixels thick (the "broaden" and "maybe-broaden" pairs). Figure 3 shows sets of pairs that are compatible with the presence of edges in various orientations.

By taking the transitive closure of mask compatibility, we obtain sets of masks that represent connected line (or edge) fragments. These are analogous to the connected components of step (b), but they are now components of masks, not of pixels, and a pixel may belong to more than one component.

We can now illustrate the advantage of keeping all the masks at each pixel rather than just the "strongest" one. [The examples used in this and the following paragraphs are all fragments of roads on an aerial photograph; they are all taken from Figure 4.] Figure 5a shows an 8 × 8 portion of the image; the numbers in the centers of the squares are the pixel gray levels. In each square we show the masks that describe the 3 × 3 neighborhood of that pixel; the number under each mask is a measure of its contrast. The lines joining the masks represent compatibilities (solid lines for "extend" compatibilities; dashed lines for "broaden" compatibilities). We see that this piece of image contains a branching thin region. If we keep only the highest-contrast mask at each pixel, we obtain Figure 5b, in which nearly all the compatibilities are gone! Things are somewhat better, but still far from perfect, if we keep the two highest-contrast masks at each pixel (Figure 5c).

We next illustrate the use of the masks in "thinning"; by analyzing the mask patterns to determine which pixels lie on or closest to the midline of the curvilinear feature. In Figure 6, (a) shows a two pixel thick linear feature; (c–d) show standard thinning results (depending on which side we thin from first); and (b) shows the result when the masks are used. The details of how this result was obtained are shown in Figure 7a, where the numbers under the masks are now component numbers; component 949 is the one shown in Figure 6. The masks show that the pixels in column 93 better represent the midline in rows 70–73, while those in column 92 better represent it in rows 74–77. In Figure 7b the masks not on the midline have been dropped (though their links to other masks have been kept).

Finally, we illustrate the use of the masks to fill small gaps. Figure 8 shows the midlines of three linear feature fragments, two denoted by black squares and one by white circles. If we regarded the fragments as sets of pixels, and filled the gap on geometric grounds, we would join the white fragment to the lower black one. Inspection of the masks, however, which are shown (for the boxed center part of Figure 8) in Figure 24, indicates that we should in fact join the white fragment (No. 240) to the upper black one (No. 647), not to the lower one (No. 626).

A more detailed discussion of the mask-based approach to curvilinear feature detection can be found in [2–4], where additional examples are also given. We believe that the mask-based approach can be useful in the precise delineation of thin linear features. It may also

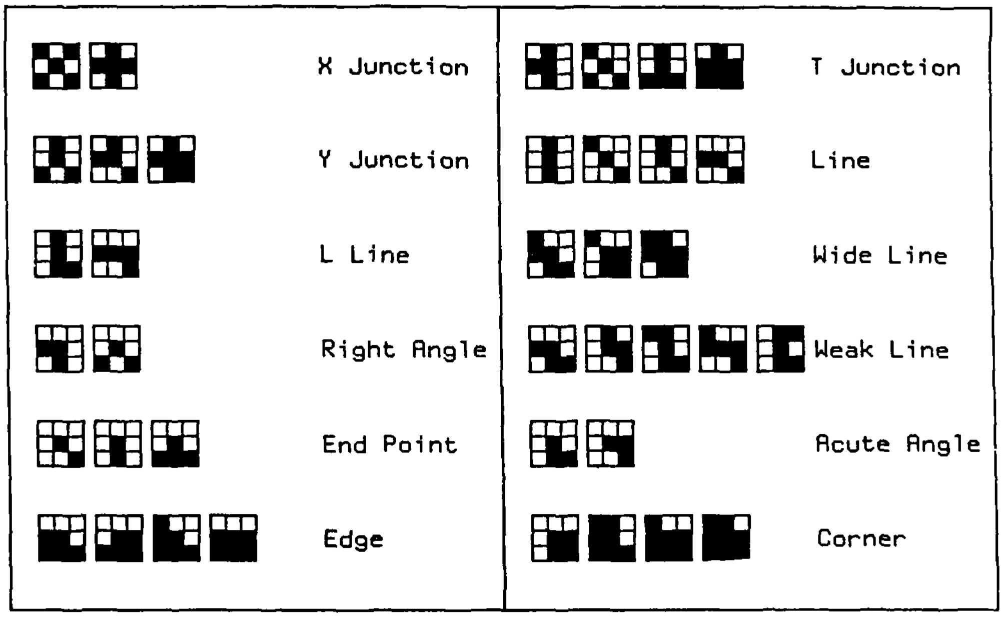


Figure 1. Local edge and line patterns.

be of value in precisely delineating the edges of thick objects; here we would use only the edge-like masks.

It should be pointed out that the mask-based approach to extracting edge or curve fragments is likely to be more computationally costly than conventional approaches. However, when massively parallel SIMD systems are available at low cost, which can be expected to happen during the next decade, the extra computational cost of detailed pixel-neighborhood analysis will no longer be a major factor. We have therefore concentrated on demonstrating the advantages of such analysis, in terms of the accuracy of the resulting linear feature delineations, on the assumption that in the relatively near future, it will no longer be necessary to rule out this approach on grounds of cost.

References

1. A. Rosenfeld and A.C. Kak, *Digital Picture Processing*, second edition, 1982, Sections 10.2-4.
2. N. Netanyahu and A. Rosenfeld, Mask matching for linear feature detection, TR-1759, Center for Automation Research, University of Maryland, College Park, January 1987.

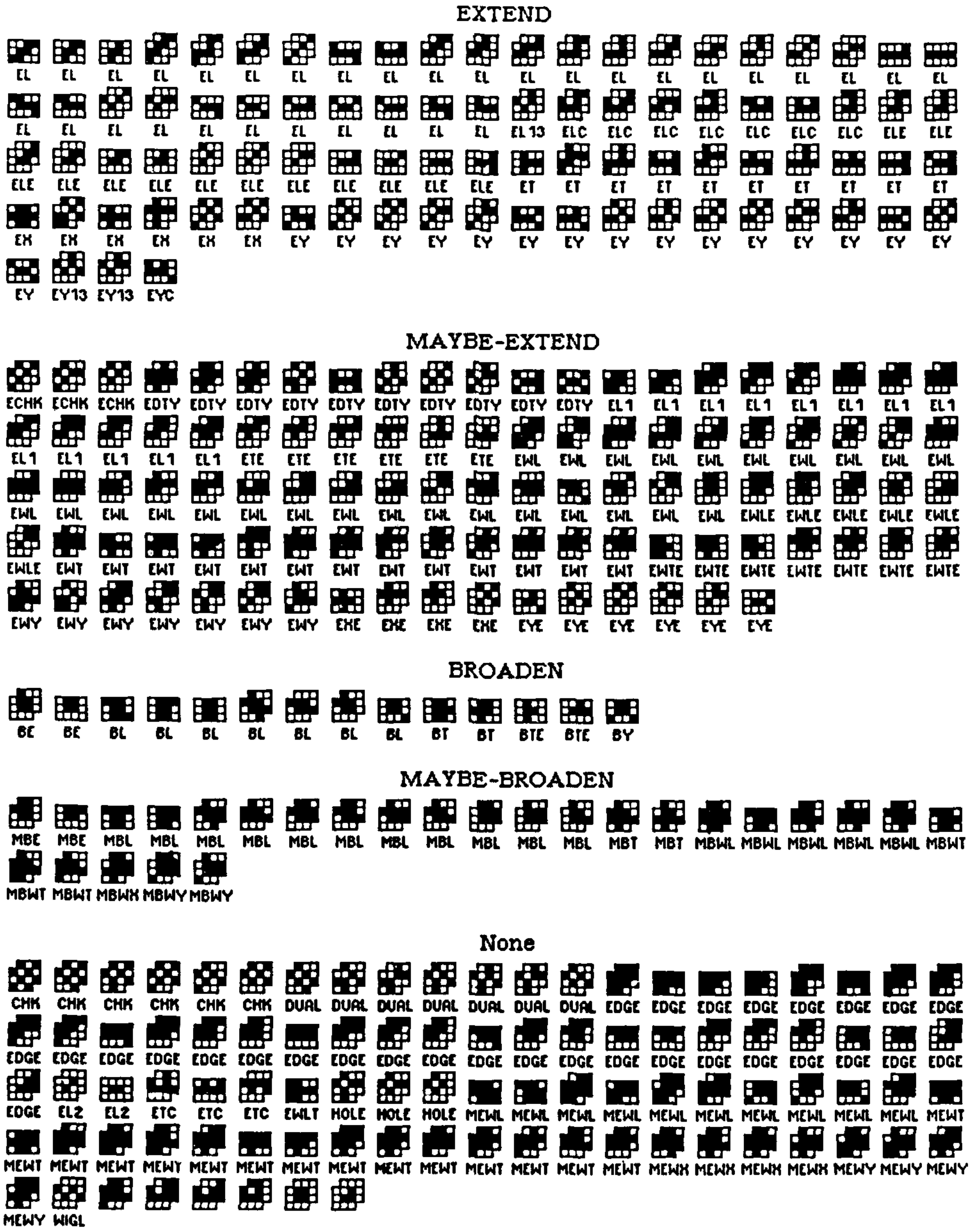

EXTEND
EL EL EL EL EL EL EL EL EL EL EL EL EL EL EL EL EL EL EL EL EL
EL EL EL EL EL EL EL EL EL EL EL EL 13 ELC ELC ELC ELC ELC ELC ELC ELE ELE
ELE ELE ELE ELE ELE ELE ELE ELE ELE ELE ELE ET ET ET ET ET ET ET ET ET ET
EX EX EX EX EX EX EY EY EY EY EY EY EY EY EY EY EY EY EY EY EY
EY EY13 EY13 EYC
MAYBE-EXTEND
ECHK ECHK ECHK EDTY EDTY EDTY EDTY EDTY EDTY EDTY EDTY EDTY EDTY EL1 EL1 EL1 EL1 EL1 EL1 EL1 EL1
EL1 EL1 EL1 EL1 EL1 ETE ETE ETE ETE ETE ETE EWL EWL EWL EWL EWL EWL EWL EWL EWL EWL
EWL EWL EWL EWL EWL EWL EWL EWL EWL EWL EWL EWL EWL EWL EWL EWL EWLE EWLE EWLE EWLE EWLE
EWLE EWT EWT EWT EWT EWT EWT EWT EWT EWT EWT EWT EWT EWT EWTE EWTE EWTE EWTE EWTE EWTE EWTE
EWY EWY EWY EWY EWY EWY EWY EXE EXE EXE EXE EYE EYE EYE EYE EYE EYE
BROADEN
BE BE BL BL BL BL BL BL BL BT BT BTE BTE BY
MAYBE-BROADEN
MBE MBE MBL MBL MBL MBL MBL MBL MBL MBL MBL MBL MBL MBT MBT MBWL MBWL MBWL MBWL MBWL MBWT
MBWT MBWT MBWX MBWY MBWY
None
CHK CHK CHK CHK CHK CHK DUAL DUAL DUAL DUAL DUAL DUAL DUAL EDGE EDGE EDGE EDGE EDGE EDGE EDGE EDGE
EDGE EDGE EDGE EDGE EDGE EDGE EDGE EDGE EDGE EDGE EDGE EDGE EDGE EDGE EDGE EDGE EDGE EDGE EDGE EDGE EDGE
EDGE EL2 EL2 ETC ETC ETC EWLT HOLE HOLE HOLE MEWL MEWL MEWL MEWL MEWL MEWL MEWL MEWL MEWL MEWL MEWT
MEWT MEWT MEWT MEWY MEWT MEWT MEWT MEWT MEWT MEWT MEWT MEWT MEWT MEWT MEWX MEWX MEWX MEWX MEWY MEWY MEWY
MEWY WIGL

Figure 2. Consistent pairs of local edge or line patterns.

3. J. Canning, J.J. Kim, and A. Rosenfeld, Symbolic pixel labeling for linear feature detection, TR-1761, Center for Automation Research, University of Maryland, College Park, January 1987.

4. J.J. Kim and A. Rosenfeld, Feature detection based on pairwise consistent labeling, TR-1792, Center for Automation Research, University of Maryland, College Park, January 1987.

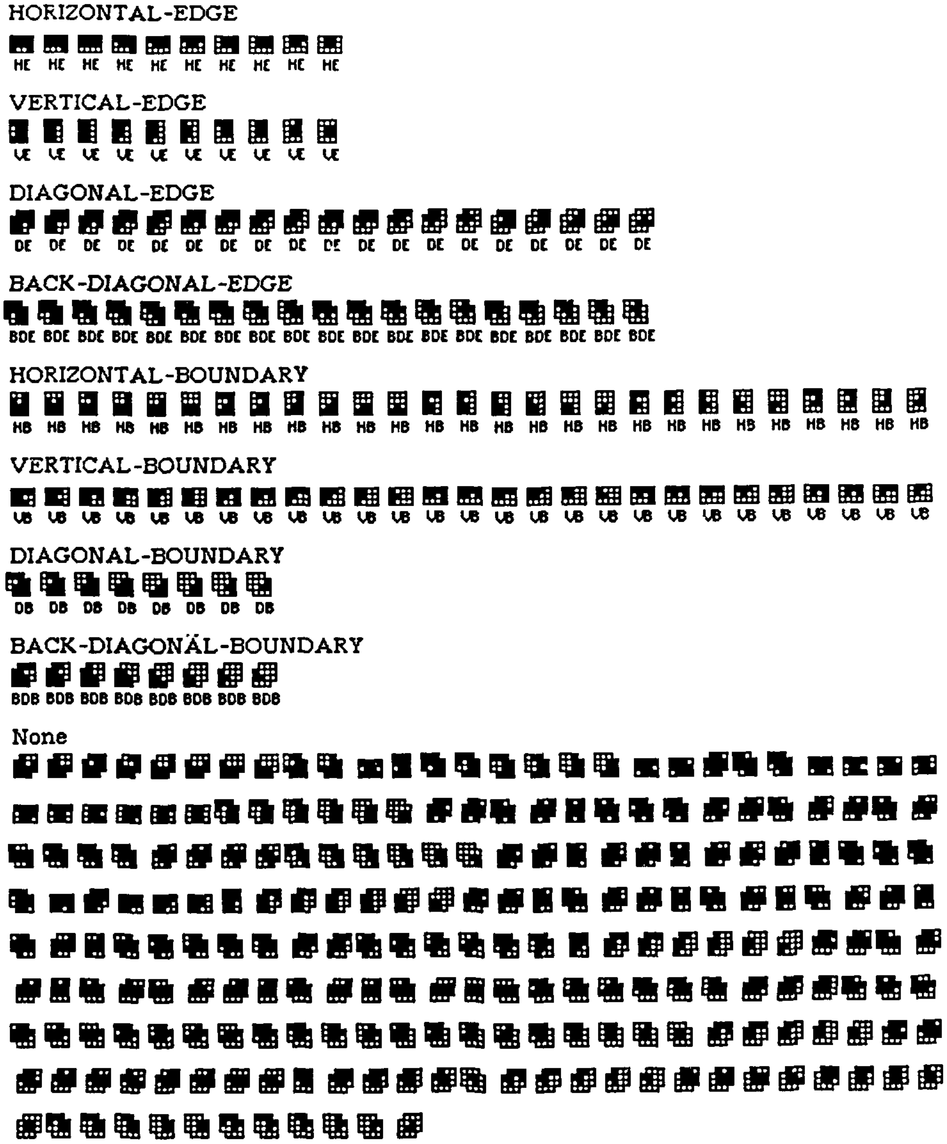

HORIZONTAL-EDGE
HE HE HE HE HE HE HE HE HE HE
VERTICAL-EDGE
VE VE VE VE VE VE VE VE VE VE
DIAGONAL-EDGE
DE DE DE DE DE DE DE DE DE DE DE DE DE DE DE DE DE DE DE
BACK-DIAGONAL-EDGE
BDE BDE BDE BDE BDE BDE BDE BDE BDE BDE BDE BDE BDE BDE BDE BDE BDE BDE BDE
HORIZONTAL-BOUNDARY
HB HB HB HB HB HB HB HB HB HB HB HB HB HB HB HB HB HB HB HB HB HB HB HB HB HB HB
VERTICAL-BOUNDARY
VB VB VB VB VB VB VB VB VB VB VB VB VB VB VB VB VB VB VB VB VB VB VB VB VB VB VB
DIAGONAL-BOUNDARY
DB DB DB DB DB DB DB DB
BACK-DIAGONAL-BOUNDARY
BDB BDB BDB BDB BDB BDB BDB BDB
None

Figure 3. Consistent pairs of local edge patterns.

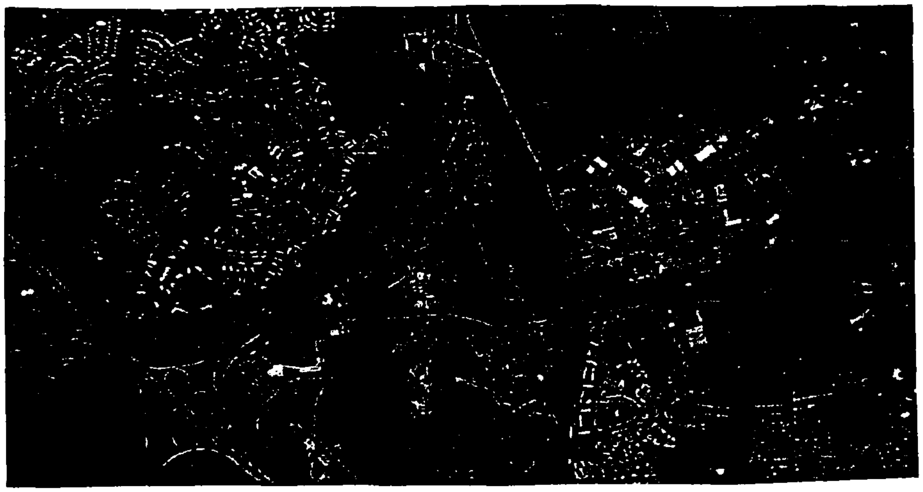


Figure 4. Roads

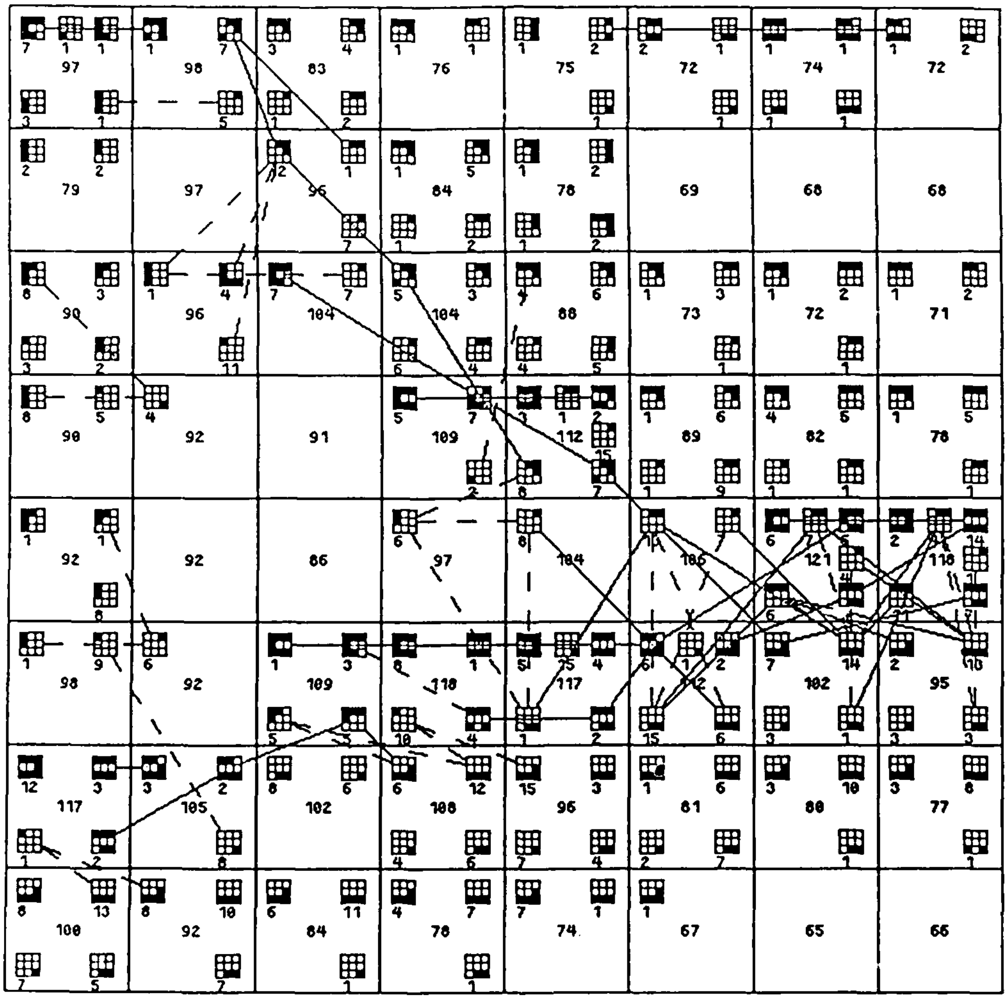
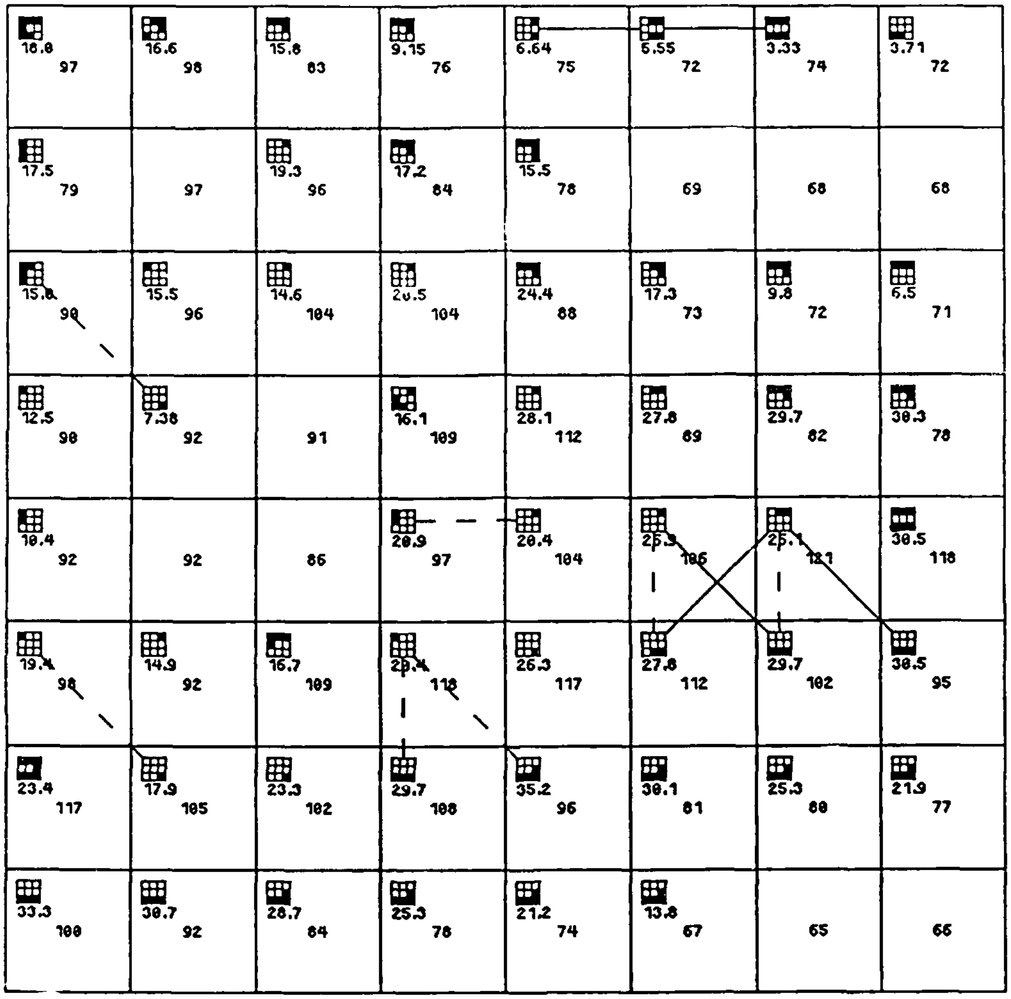


Figure 5a. An 8 × 8 image showing masks and compatibilities.



10.0 97	16.6 98	15.8 83	9.15 76	6.64 75	6.55 72	3.33 74	3.71 72
17.5 79	97	19.3 96	17.2 84	15.5 78	69	68	68
15.0 90	15.5 96	14.6 104	20.5 104	24.4 88	17.3 73	9.8 72	6.5 71
12.5 90	7.38 92	91	16.1 109	28.1 112	27.8 69	29.7 82	30.3 78
10.4 92	92	86	20.9 97	20.4 104	25.3 106	25.1 121	30.5 118
19.4 98	14.9 92	16.7 109	20.4 118	26.3 117	27.8 112	29.7 102	30.5 95
23.4 117	17.9 105	23.3 102	29.7 108	35.2 96	30.1 81	25.3 80	21.9 77
33.3 100	30.7 92	28.7 84	25.3 78	21.2 74	13.8 67	65	66

Figure 5b. Results of keeping only the strongest mask at each pixel.

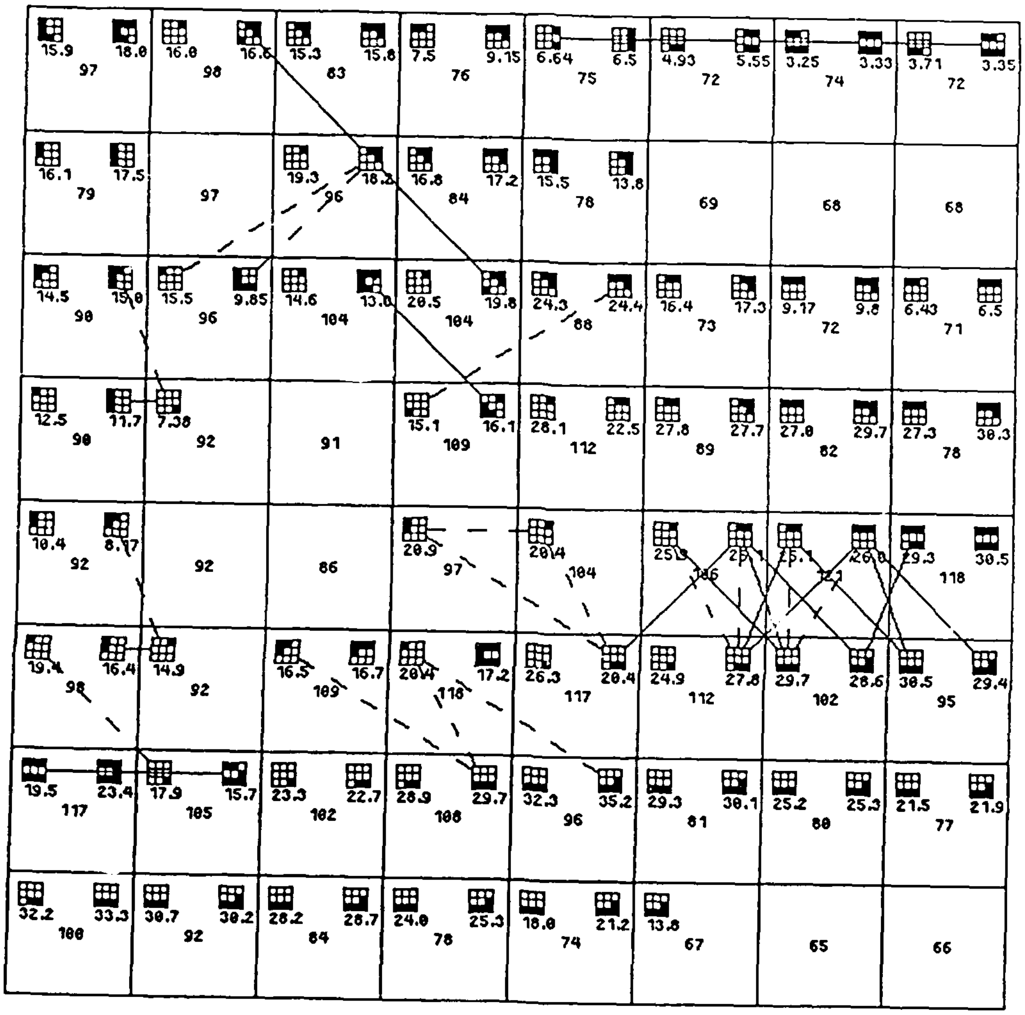
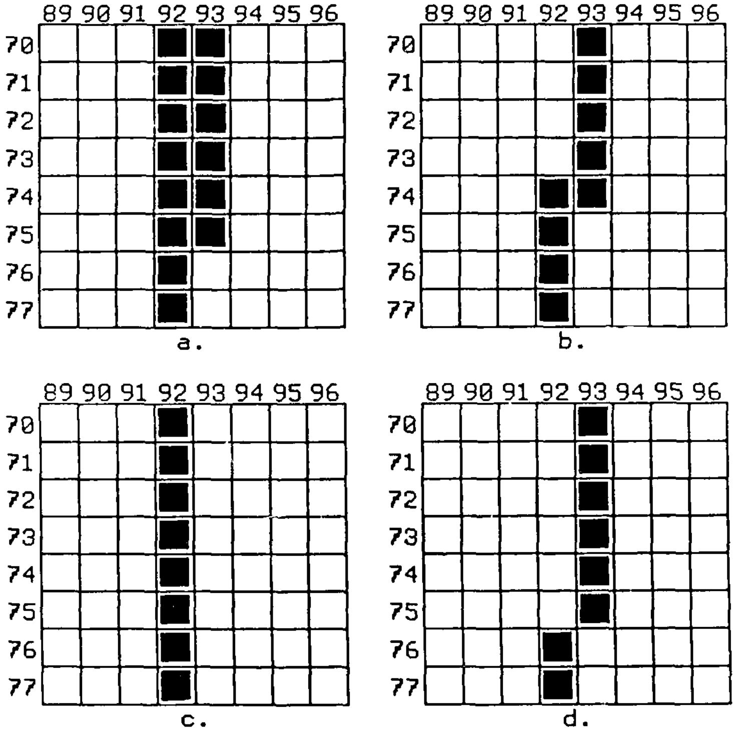


Figure 5c. Results of keeping only the two strongest masks at each pixel.



	89	90	91	92	93	94	95	96
70				■	■			
71				■	■			
72				■	■			
73				■	■			
74				■	■			
75				■	■			
76				■				
77				■				

a.

	89	90	91	92	93	94	95	96
70					■			
71					■			
72					■			
73					■			
74				■	■			
75				■				
76				■				
77				■				

b.

	89	90	91	92	93	94	95	96
70				■				
71				■				
72				■				
73				■				
74				■				
75				■				
76				■				
77				■				

c.

	89	90	91	92	93	94	95	96
70					■			
71					■			
72					■			
73					■			
74					■			
75					■			
76				■				
77				■				

d.

Figure 6. (a) Fragment; (b) midline found using masks (see Figure 7); (c–d) results of conventional thinning (from the right and left, respectively).

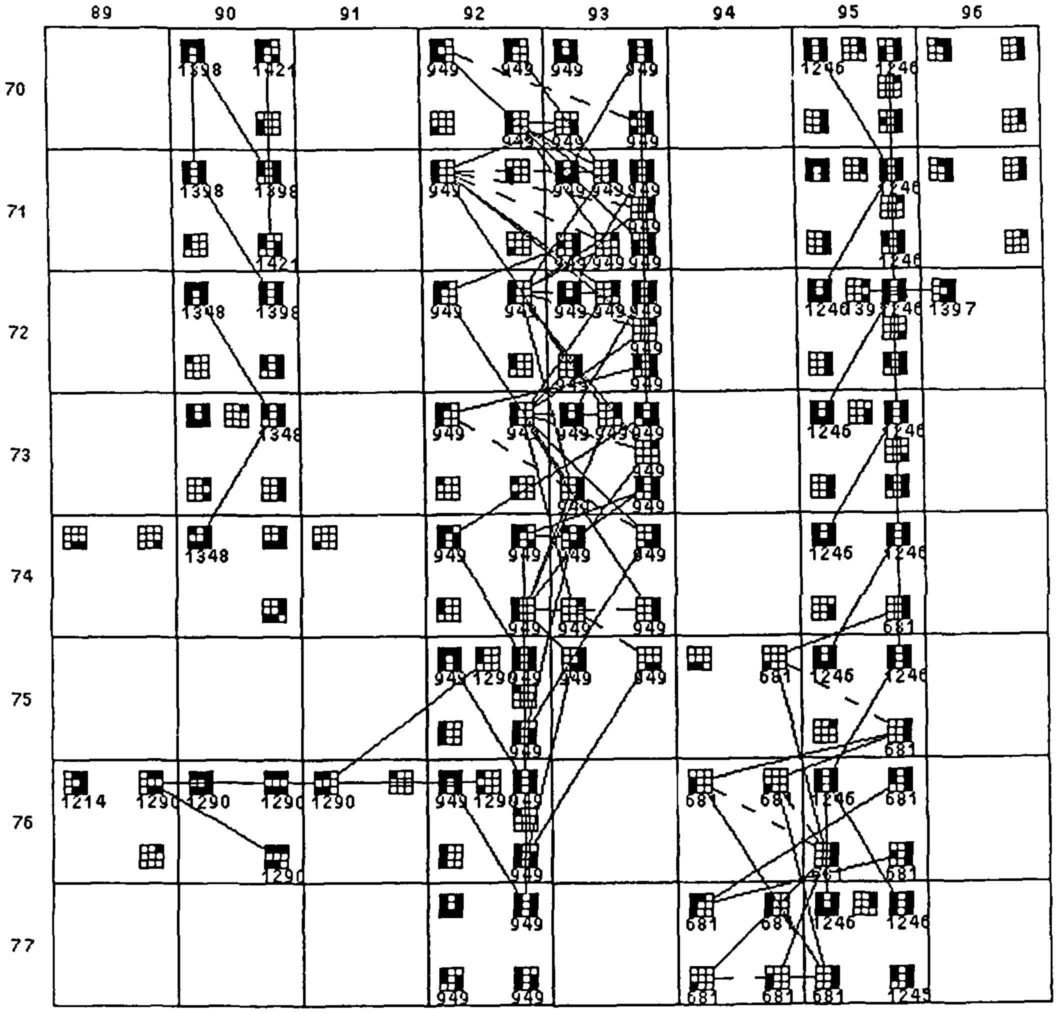


Figure 7a. Masks; the fragment in Figure 6 is No. 949.

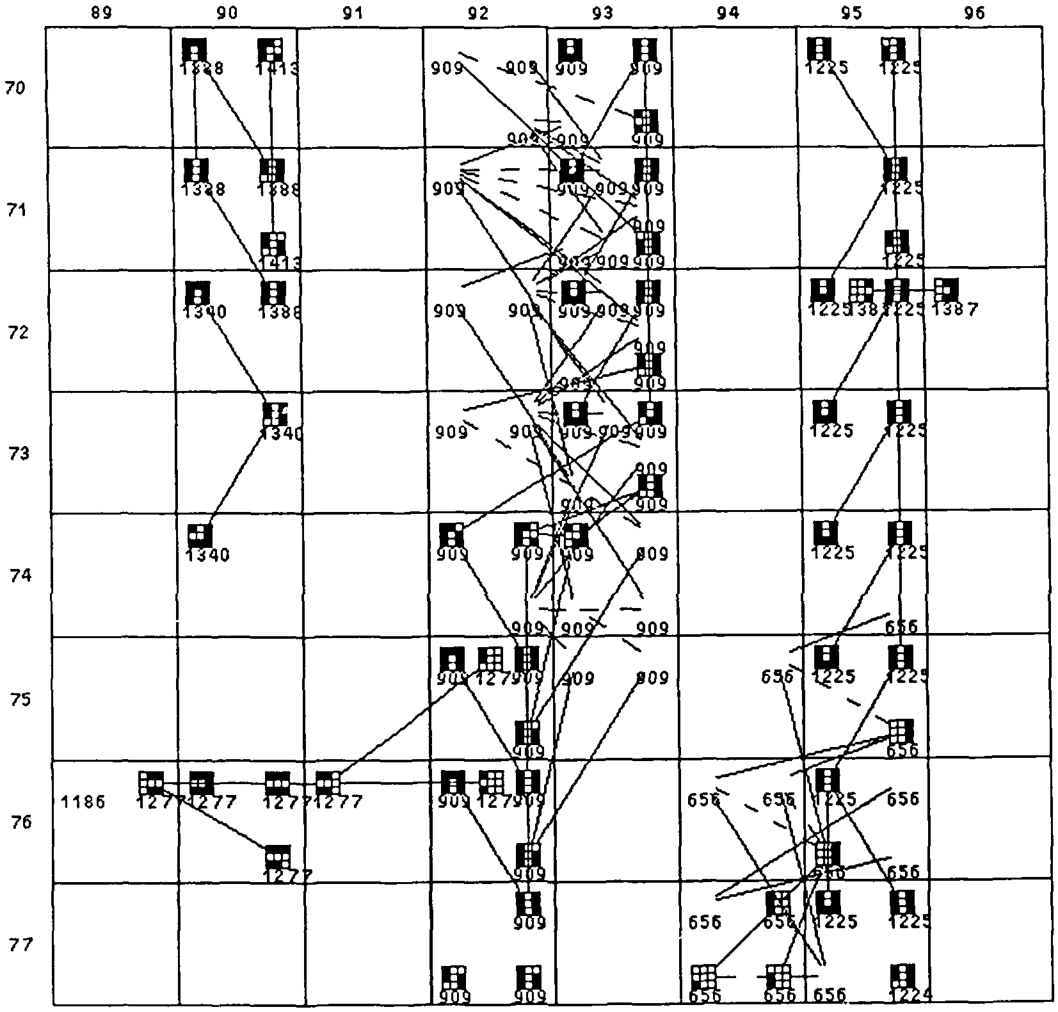


Figure 7b. Masks not on the midline have been deleted (though the links to them are still shown).

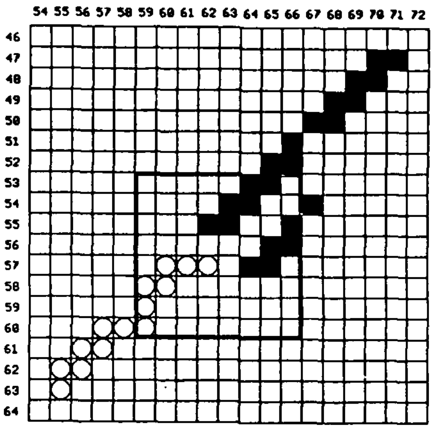


Figure 8. Three fragments that could be linked.

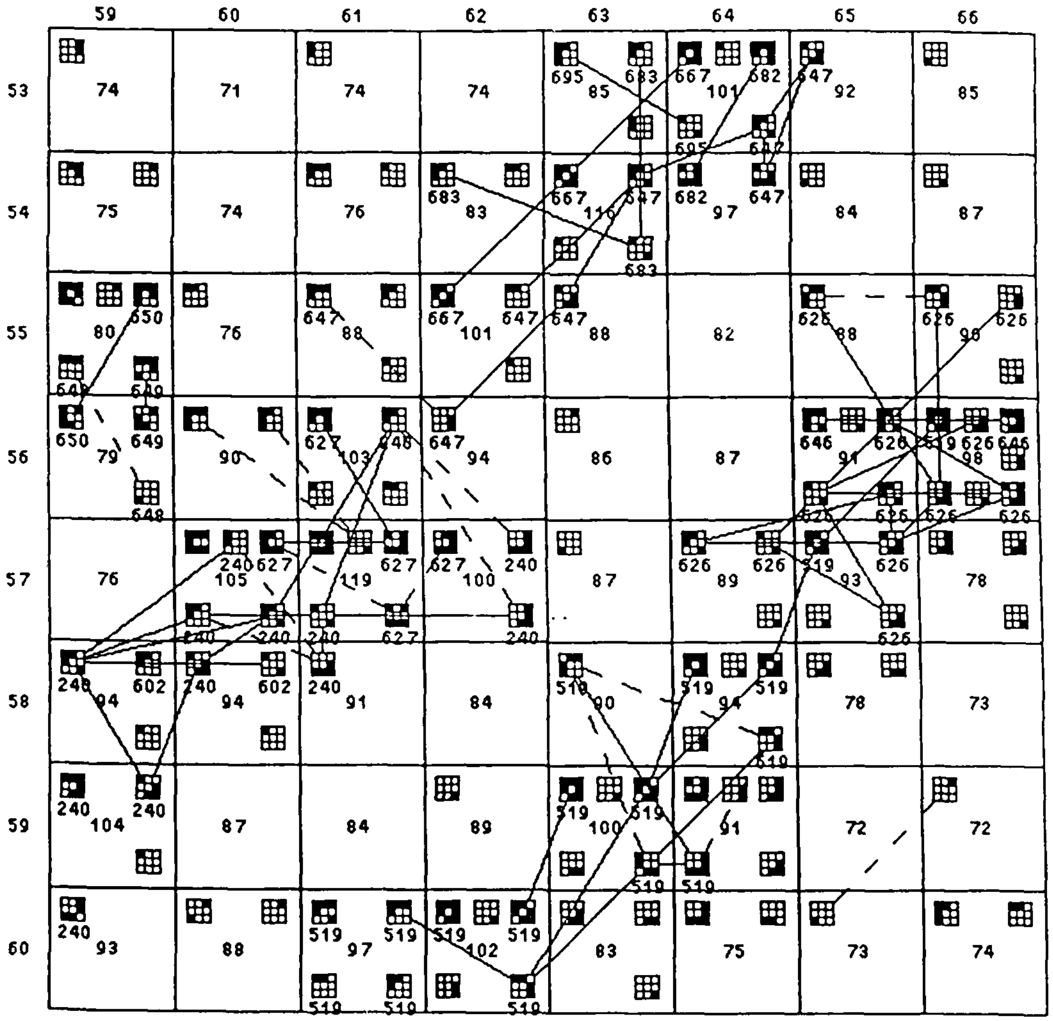


Figure 9. Masks for the center part of Figure 8.

Image Segmentation Using Geometric and Physical Constraints *

Thilaka S. Sumanaweera, Glenn Healey, Byung-Uk Lee,
Thomas O. Binford and Jean Ponce

Robotics Laboratory
Computer Science Department
Stanford University
Stanford, California 94305

Abstract

Segmentation is an important step in scene analysis. Segmentation algorithms based on geometry alone are easily confused by image artefacts due to physical processes, for example, specularities. On the other hand, segmentation algorithms based on purely local properties are rarely adequate for segmenting complex images. In this paper, we describe segmentation algorithms which use both geometric and physical properties. We discuss the performance of these algorithms.

Introduction

Segmentation is a fundamental first step in scene analysis. Good edge detectors, such as [Canny, 1986], and [Nalwa, 1986], have recently become available. However, there is still a lack of robust algorithms to identify geometrically and physically meaningful regions and contours in images which are actual projections of three-dimensional surface patches and edges.

In this paper, we describe several implemented segmentation algorithms which use either geometric or physical constraints. This paper is organized into six sections. Sections 1, 2, and 3 describe our geometric segmentation algorithms. These algorithms rely on image symmetries and graphs of regions, edges, and edge junctions to find meaningful regions in the image. Sections 4,5, and 6 describe our physical segmentation algorithms. Section 4 provides an overview. Section 5 presents an algorithm which uses physical constraints to link edges. Section 6 discusses the problem of constructing a scene description in terms of surface patches, their properties, and their relationships.

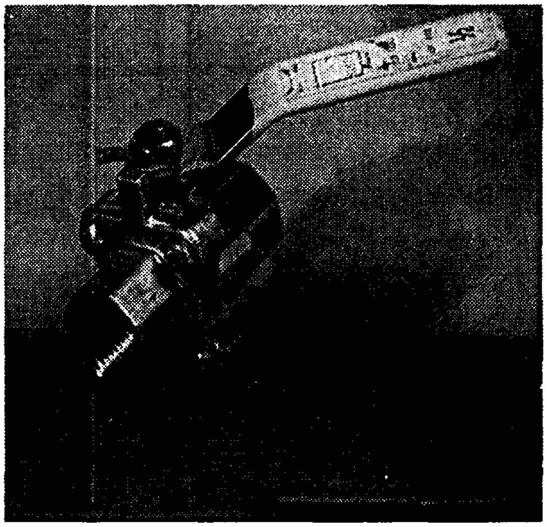


Figure 1. The test image: a Smith valve.

1. Geometric segmentation

In the next three sections, we concentrate on geometric segmentation using contour information. We assume that edges have been found and linked using some edge detection scheme (e.g., [Canny, 1986],[Nalwa, 1986],[Nalwa and Pauchon, 1987]). We don't assume, however, perfect linking, closed contours, or perfect junctions. On the contrary, we assume that edges may be wrongly linked, that gaps and edge terminations exist, and that junctions may be missed.

To use this imperfect boundary information, it is necessary to characterize relations between edge fragments. We consider in this paper two types of such relations, *oppositeness*, and *curvilinearity*. Oppositeness relates different edges which bound a given region. We give in section 2 an algorithm to find ribbons, which correspond to an oppositeness relation. Curvilinearity

*Support for this work was provided in part by the Air Force Office of Scientific Research under contract F33615-85-C-5106 and by the Advanced Research Projects Agency of the Department of Defense under Knowledge Based Vision ADS subcontract S1093 S-1.

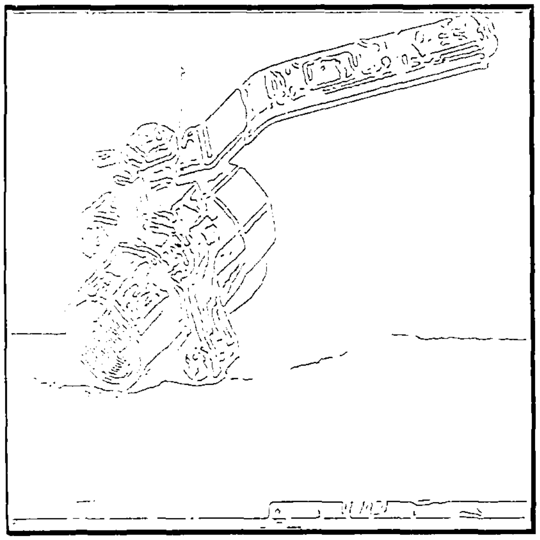


Figure 2. The edges found in the valve image using Canny's [Canny, 1986] edge detector.

relates edges which satisfy a geometric continuity condition; it extends colinearity. In section 3, we give an algorithm, based on curvilinearity, for building a graph structure of edges separated by junctions, and finding cycles in this graph.

Both algorithms have been implemented on a Symbolics lisp machine. They are demonstrated on a 256 x 256 image of a *Smith Valve*, shown in Figure 1. We have used an implementation of Canny's edge detector [Canny, 1986] to provide the initial edge fragments. Figure 2 shows the edges output by Canny's operator.

2. Segmentation into ribbons

In this section, we describe an algorithm for finding ribbons in an image from contour data. This algorithm is based on Nevatia's and Binford's method of projections [Nevatia and Binford, 1977], but presents several significant improvements. In particular, our algorithm deals with imperfect contours and different types of ribbons.

We first define ribbons (section 2.1), summarize the original Nevatia and Binford algorithm, and compare it to different methods (section 2.2). We then detail our algorithm (section 2.3), and illustrate and discuss its performances on some examples (section 2.4). Finally (section 2.5), we discuss our plans for future work. We give rough estimates of the time complexity of the different steps of our algorithm throughout.

2-1. Ribbons

In this paper, a ribbon is the plane shape generated by sweeping a line segment, called a generator, or cross-section, along a plane curve, called an axis, or spine. The oppositeness relation used here is the relation between the sides of a ribbon.

In [Rosenfeld, 1986], Rosenfeld discusses more general classes of ribbons, where the generator can be an arbitrary plane shape, and calls the class of ribbons considered here L-ribbons (for line ribbons). See also [Ponce, 1988] for an extension of Rosenfeld's results.

The length of the generators, as well as their orientation with respect to the spine, may vary. Two important classes of ribbons are the so-called Brooks and Brady ribbons, following Rosenfeld's terminology [Rosenfeld, 1986].

Brooks ribbons are ribbons such that the angle between the generators and the tangent to the spine is constant. They are two-dimensional generalized cylinders [Binford, 1971]. In Acronym, Brooks [1981] restricts himself mainly to ribbons with a straight spine.

Brady ribbons (also called smooth local symmetries) [Brady and Asada, 1984] are ribbons such that any generator makes equal angles with the sides of the ribbon, i.e., the ends of the generators are locally symmetric with respect to their mid-points.

2-2. The method of projections

Assume for the moment that it is possible to decide whether two contour points form a ribbon pair (i.e., are the end-points of a ribbon generator), based on some image measure. This can clearly be done for Brady ribbons (local symmetries are characterized by the fact that the angle between a generator and the outward normals at its ends are the same).

We will come back later to this problem for other types of ribbons. Under the above assumption, ribbons can be found in an image by first finding ribbon pairs, and then grouping ribbon pairs into ribbons. The axis of a ribbon is given by the mid-points of its generators.

The first part of this algorithm, finding ribbon pairs, can be implemented in different ways, more or less efficient. The simplest implementation of this algorithm is to test all possible pairs of points. This is the algorithm used in Brady and Asada [1984] to find Brady ribbons. Its complexity is $O(n^2)$, where n is the number of contour points (Brady and Asada also propose a much faster algorithm, based on analytical solutions for local symmetries between circular arcs, but it does not generalize to other types of ribbons).

Another implemented algorithm is the method of projections, first proposed by Nevatia and Binford [1977]. It can be summarized as follows: discretize the possible orientations of local ribbon axes; for each of

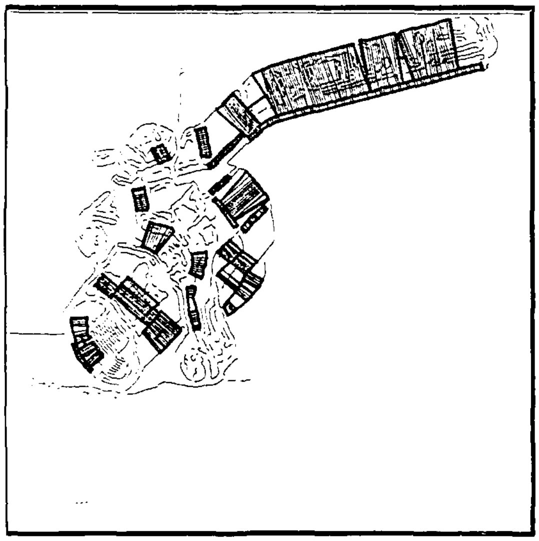


Figure 3. Some ribbons found by the algorithm of section 2.3.

these directions, project all contour points into buckets, and verify that pairs of points within the same bucket form ribbon pairs by test of their angular bisector. The complexity of this method is $O(k.n)$, where k is the number of discretized orientations. For large numbers of contour points, the second method is clearly more efficient.

2-3. Algorithm

The original method of projections assumes closed contours, and works only for the class of ribbons considered by Nevatia and Binford. To extend this method to deal with different kinds of ribbons, and imperfect contour data, we use the following algorithm, based on a cost function that is zero if a pair of points is a perfect ribbon pair, and increases as the pair of points becomes a poorer approximation to a ribbon pair.

The algorithm can be decomposed into three steps as follows:

1. **Projecting points.** For each discretized projection direction, do: project all contour points in buckets; compare pairs of points within each bucket, and assign a cost to each of these pairs; output each pair whose cost is below a given threshold.

2. **Sorting ribbon pairs.** Initialize a current list of ribbon pairs to nil; for every ribbon pair output by the previous step, add the pair to the current list of ribbon pairs, keeping it sorted according to the cost function.

3. **Building ribbons.** Repeat the following until the list is empty: pop the list, the corresponding pair of points is a new ribbon pair; if it is connected to an existing ribbon, add the pair to the ribbon, otherwise, create a new ribbon; deactivate all incompatible ribbon pairs.

Some details are missing in the previous algorithm. The first step of the algorithm is relatively straightforward, although it is important to ensure that buckets are large enough to avoid missing some ribbon pairs (i.e., buckets must overlap). It should be noticed that in the original Nevatia and Binford algorithm, the complexity of this step is $O(k.n)$, because each edge point belongs to at most one ribbon pair per discretized orientation. Here, as we don't have closed contours, a given point may form ribbon pairs with any other point in his bucket which belongs to another contour fragment. Thus, the complexity is roughly multiplied by the number e of edges and becomes $O(e.k.n)$, which is still better than $O(n^2)$, especially in high resolution images.

The rest of the algorithm is independent of the method of projections, and could be used as a post-processing stage to the Asada and Brady [1984] algorithm as well. In the second and third steps of the algorithm, it is important to represent efficiently the current list of ribbon pairs, so that insertion and deletion of a given element, as well as popping the list, can be done efficiently. We have chosen to represent this list by an array of buckets, indexed by the cost function. Each bucket itself is a conventional doubly linked list, sorted according to the cost function.

This data structure allows insertion of new pairs in almost constant time (finding the right bucket is done in constant time, insertion in the doubly linked list corresponding to the bucket depends on the lists's length, which is in general bounded). Deletion is done in constant time by using the doubly linked lists. Popping the list is also done in constant time. This implies that the complexity of steps 2 and 3 is proportional to the number of ribbon pairs generated, so it is also roughly $O(e.k.n)$. This justifies again the use of the method of projections.

Two ribbon pairs are neighbors iff corresponding points in the pairs are neighbors. For each edgel, we build a list of adjacent ribbon pairs during the projection step. In particular, when a ribbon pair rp is popped from the current list, the incompatible ribbon pairs are the pairs rp_i's whose both ends are in a given neighborhood of the ends of rp, and are such that rp and rp_i cross each other. Finding the ribbon pairs incompatible with a given ribbon pair is straight forward. Our algorithm solves the problem of choosing between local ribbon descriptions by ensuring that the "best" ribbon pairs are built first, forming seeds for larger ribbons. Notice however that a given region can be described by several ribbons, reflecting the different symmetries of that region.

2-4. Results

The algorithm has been implemented for Brady ribbons. In this case, the choice of a cost function is straightforward. For example, let α_i be the angle between the line joining the end-points $p_i(i = 1, 2)$ in a ribbon pair and the outward normal at p_i, the cost function can be defined as $|\alpha_2 - \alpha_1|$. We have added a last step to the segmentation algorithm. The previous algorithm assumes that ribbons are made of connected regions of ribbon pairs. In real images, there may be gaps between ribbon pairs. Thus, after all elementary ribbons have been found, one more grouping step is performed to merge elementary ribbons into larger ribbons, based on a distance measure between the ribbon-ends (the ends of a ribbon are ribbon pairs).

Some of the results of segmentation on the test image are shown in Figure 3. Facets of the valve and its handle have been segmented by this technique. The algorithm found most regions which a human would identify as being symmetric. There were a few false positives. The whole program took 1.5 hours on a Symbolics LISP machine. 75% of this time was spent to find ribbon pairs, to discard inconsistent ribbon pairs, and to group them into ribbons. Only 25% of the time was spent in projecting contour points. The situation would be inverted if the $O(n^2)$ algorithm had been used, resulting in a much longer computing time.

2-5. Future work

In the implemented algorithm, we have only considered Brady ribbons. The algorithm clearly generalizes to any other type of ribbons such that some cost function can be assigned to any potential ribbon pair based on local measures. In their original algorithm, Nevatia and Binford [1977] do not use a cost function, but consider pairs of adjacent ribbon pairs, and reject the pairs such that the angle α between the line joining their mid-points and the generators is too far from 90 degrees. Instead of using these pairs of ribbon pairs, we plan to find right Brooks ribbons by replacing edgels by data structures pointing to adjacent pairs of edgels (some kind of "mid-edgels"), and using them to generate ribbon pairs, with a cost function based on the angle α. Unfortunately, this method does not work for arbitrary Brooks ribbons, for which the angle between generators and spine is unknown. In [Ponce, 1988], local characterizations of some other classes of Brooks ribbons, such as worms and skewed symmetries [Kanade, 1981], are described. We plan to implement them in the near future.

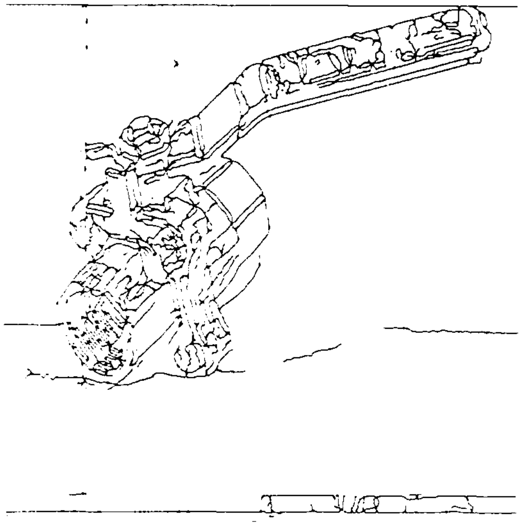


Figure 4. The junctions found by the algorithm described in section 3.2.

3. Graphs of junctions and edges

In this section, we describe an algorithm for building an edge-graph from imperfect contour data; nodes in the graph are edge junctions, arcs are edges between junctions. An application of this algorithm is in finding geometrically meaningful regions as cycles in this graph. The basic idea here is to use boxes enclosing contour segments to determine the curvilinearity of contour fragments and find junctions.

In section 3.1, we briefly describe an algorithm for segmenting contour fragments into line segments, and enclosing the contours into rectangular boxes. These boxes and a quadtree representation of images are used in section 3.2 to find contour junctions and build the graph representation. This graph is used in section 3.3 to find minimal cycles. Section 3.4 shows and discusses some results.

3-1. Linear approximation

Robust algorithms for segmenting contours into line segments are useful for many tasks, including recognition of two- or three-dimensional objects in images (e.g., [Ayache and Faugeras, 1986], [Grimson and Lozano-Perez, 1987], [Lowe, 1987]). In this section, we briefly discuss a new algorithm which segments a contour into a set of line segments by using a bottom-up (merge) approach. We use the algorithm in the following two sub-sections to find edge junctions.

Locally, the direction of each edgel is given by the edge detector (e.g., in Canny's case [Canny, 1986], by the gradient direction). Adjacent edgels with directions close to each other can be merged to form seeds

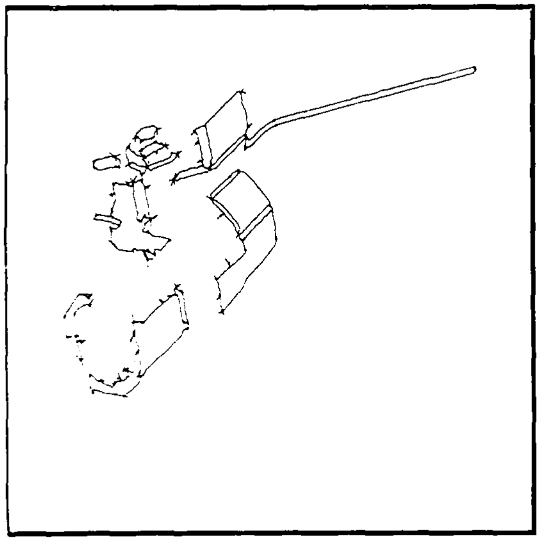


Figure 5. Some minimal cycles found by the algorithm of section 3.3.

for straight lines. Least squares linear appoximations are computed for these seeds. Adjacent seeds can in turn be merged into longer line segments if the resulting error is small enough. By repeating this process, we obtain a set of line segments approximating the contour.

The process of merging line pairs is analogous to the construction of ribbons from ribbon pairs, as described in section 2.3. We use a similar algorithm and data structure. We consider one edge fragment (i.e., a simply connected sequence of edgels) at a time. Line pairs are sorted according to the error committed when merging them into a single line. When a line pair is popped, the two corresponding lines $l_i (i = 1, 2)$ are merged into a new line. All line pairs containing the lines l_i's are displaced within the list of line pairs after replacing the corresponding l_i in each pair by the new line l, and computing the new cost of the line pair.

3-2. Building the graph

The previous algorithm provides, in addition to the line segments, rectangular boxes centered around them and enclosing the corresponding contour fragments: the width of these boxes is computed by taking into account the least squares error and the uncertainty in the edge localization (a function of edge operator size and contrast); the length of the boxes is given by the length of the line segments plus twice the localization uncertainty. The representation obtained is analogous to Ballard's strip trees [Ballard, 1981], although Ballard uses a top-down (split) approach to segmentation.

Boxes can in turn be used to find edge junctions. A junction between edges is defined as the center of gravity of the intersection of boxes. To localize intersections between boxes, we use a quadtree whose leaves are empty or point to the list of boxes intersecting them. Thus, we need only to compare a given box to the other boxes associated to the leaves it intersects.

Edges are split into shorter edges at junctions, which form nodes in a graph. An arc in that graph is labelled by the corresponding edge, and relates the junctions at the ends of that edge. Notice that more than two edges may be incident at a given junction (T-junctions: they are obtained for example when the box associated to one edge intersects the middle part of an other edge), but an edge has at most two junctions.

3-3. Finding cycles

The graph found in the previous section may have several connected components (e.g., the graph in Figure 4 has eight connected components). In this section, we give an algorithm to find all the minimal cycles of a given component.

A minimal cycle is a cycle which does not enclose any other cycle in the associated connected component. Minimal cycles are found by using the left shoulder algorithm [Minsky, 1969]: start at a given junction, and traverse the graph by always turning left as much as possible at each junction; if an edge termination is found, backtrack to the previous junction and continue; stop when back at the original junction. The sequence of edges followed is a minimal cycle.

By repeating this process for all junctions, all minimal cycles are guaranteed to be found. Moreover, any edge (arc) in the graph can be traversed in two directions, and may belong to at most two minimal cycles. By marking the visited edges and the directions in which they are visited, it is possible to avoid looking for cycles along edges which have already been visited.

For each cycle found, we also calculate during the construction of the cycle its angle of turn, obtained by integrating the turns made from edgel to edgel along the cycle. This allows us to categorize the area enclosed by the cycle: if the angle of turn, measured counterclockwise, is -2π, then the enclosed area is the interior of the cycle; if the angle of turn is 2π, then it is the exterior that is enclosed by the cycle. Each connected component can have at most one cycle enclosing the exterior, and an unlimited number of cycles enclosing the interior.

3-4. Results

The graph found for the Smith valve image is shown in Figure 4. Notice the junctions which had been missed in Figure 2. Some of the cycles found by our algorithm are shown in Figure 5. In this case, there are 190 cycles in the graph structure. Using a compactness measure, 23 cycles are left, they sensibly correspond to facets of the valve. The algorithm fails to segment the handle of the smith valve as one region due to the large number of cycles within this area. Note that the handle had been segmented as one region by the ribbon finder (Figure 3). The cycle finding algorithm has however found the most important facets used by Binford et al. [Binford 1987] for prediction and matching. Building the graph and finding the minimal cycles takes about 15 minutes on a Symbolics LISP machine. Therefore, this method is faster than the ribbon finding approach.

4. Physical segmentation

In addition to segmentation based on image geometry constraints, it is also possible to use constraints associated with the physics of image formation to guide segmentation. Physical constraints can often be used to resolve ambiguities that exist when considering only geometrical information.

In the next two sections we discuss segmentation algorithms based on the physics of image formation. In section 5, we present an algorithm for linking edges based on color. In section 6, we assume the result generated by our geometric segmentation algorithm to construct a *scene description* in terms of surface patches, their properties, and their relationships.

5. Edge Linking Using Color

There has been progress on edge linking algorithms using edgel proximity and direction [Nalwa, 1987]. Unfortunately, this algorithm is local in character and sometimes fails to find a mean'ngful object boundary. It has been shown that color normalized by intensity is relatively stable under changes in geometry [Healey, 1987]. Thus, normalized color information can be used to determine whether or not an edge is a material boundary. In this section, we describe an algorithm for linking edges based on normalized color. Some results are presented in 5-2.

5-1. Bivariate normal color model

We assume that intensity is the arithmetic average of the red(R), green(G) and blue(B) components and that the three primaries divided by intensity are relatively independent of surface orientation or illumination. The Karhunen-Loeve transformation was used to find principal components in normalized r, g and b. We define two components of normalized color, C_1 and C_2, by

$$C_1 = \frac{r-b}{2}$$

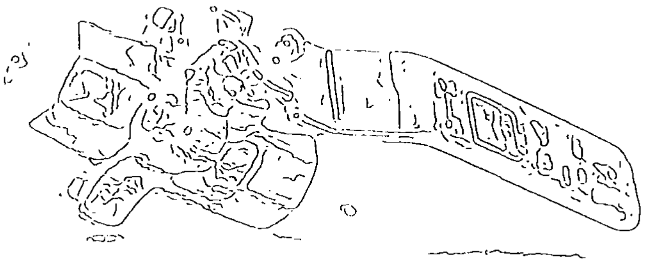


Figure 6. Edge image

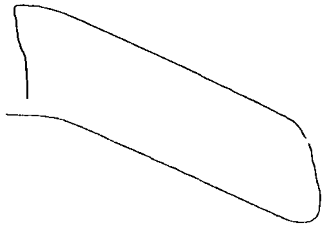


Figure 7. Linked edges for valve handle

$$C_2 = \frac{2g-r-b}{4}$$

where

$$I = \frac{R+G+B}{3}, \quad r = \frac{R}{I}, \quad g = \frac{G}{I}, \quad b = \frac{B}{I}.$$

Intuitively, C_1 is an odd spectral component and C_2 is an even component. The normalized color components of a pixel (r g b) lie on the $r+g+b=3$ plane, and it is easy to verify that the C_1 and C_2 vectors lie on the plane and are perpendicular to each other.

Given an image edge, we can take sets of pixel samples from each side of the edge. We approximate the distributions of C_1 and C_2 over the samples on one side of an edge by the bivariate normal distribution

$$F(C_1,C_2) = \frac{1}{2\pi\sigma_1\sigma_2\sqrt{1-\rho^2}}\, e^{-U(C_1,C_2)/2}$$

where,

$$U(C_1,C_2) = \frac{1}{1-\rho^2}\left[\frac{(C_1-\mu_1)^2}{\sigma_1^2} + \frac{2\rho(C_1-\mu_1)(C_2-\mu_2)}{\sigma_1\sigma_2} + \frac{(C_2-\mu_2)^2}{\sigma_2^2}\right]$$

and μ_1, μ_2 are the means, and σ_1, σ_2 are the standard deviations of C_1 and C_2 respectively. ρ is the correlation coefficient. The two normalized color components

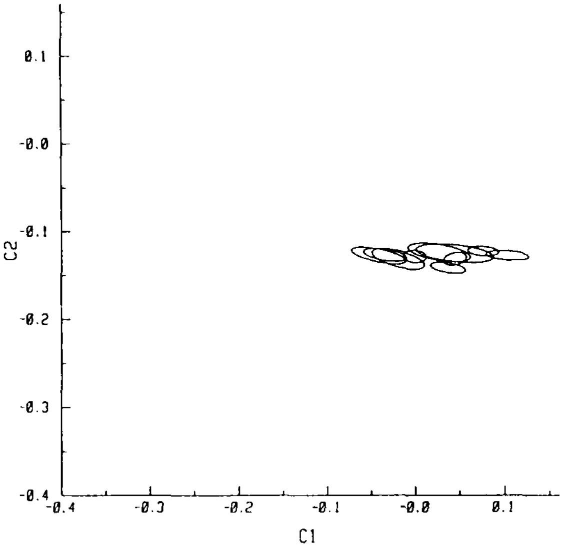


Figure 8. See text

and the bivariate normal approximation have been used successfully to segment outdoor scenes [Kong 1985].

5-2. Experimental Results

An implementation of Canny's edge detector [Canny, 1986] was used on a color image of a valve part. The edges were subsequently divided into linear segments. Pixels along the perpendicular direction to an edgel were sampled on each side of the edgel. Pixels within two pixels of an edgel were not sampled to avoid the color blurring which can occur across edgels. This guard size of two pixels was found to be reasonable for the valve image. The sample and guard size can be adaptively changed to avoid sampling across a neighboring edge.

Parameters for the bivariate normal distribution were calculated from the samples. The Mahalanobis norm was used to define the distance between sample means. The covariance matrix was calculated over the entire image except at black pixels.

Let, μ_a and μ_b be the mean vectors of colors of neighboring edges.

$$\mu_a = (\mu_{1a}\ \mu_{2a})$$
$$\mu_b = (\mu_{1b}\ \mu_{2b}).$$

Then the Mahalanobis distance d between the two mean vectors is

$$d = (\mu_a - \mu_b)\Sigma^{-1}(\mu_a - \mu_b)^t$$

where, Σ is the covariance matrix over the image.

Edgels near the end point of an edge are chosen, and the Mahalanobis distance is used to select the best edge for linking.

The edges for the valve image are shown in fig. 6, and a set of linked edges defining the yellow plastic handle is shown in fig. 7. Ellipses with $U(C_1, C_2) = 1$ for the bivariate normal approximation of the linked edge color are shown in fig. 8.

Performance of the linking algorithm is strongly influenced by the output of the edge detector. The ability of an edge detector to locate T junctions is crucial in linking.

With the correction of camera response nonlinearity, we expect that the stability of the normalized color and the linking algorithm will be enhanced.

6. Generating Physical Descriptions

Given the geometric segmentation of an image into edges and regions, it is desirable to compute physical descriptions of the surface patches producing image regions and to generate hypotheses about the physical causes of the image edges separating regions. Inferring this kind of information allows us to go from our geometrical description of the image to a physical description of the 3D scene. In this section, we discuss an approach to the problem of using our geometric segmentation to generate a description of the scene in terms of surface patches, their properties, and their relationships.

It is useful for segmentation to separate the intensity and spectral attributes of the light reflected from a surface. This is because intensity provides strong information about geometry, while color provides strong information about material composition [Shafer. 1984], [Healey, 1987]. Since one of the goals of physical segmentation is to distinguish geometrical variation from material variation in the scene, computing distinct intensity and color descriptions for each pixel in the image is an important step towards physical segmentation. In this work, we use the method described in [7] to compute distinct intensity and color descriptions for each pixel in an image region.

6-1. Relating Surface Patches

In this subsection, we examine the problem of inferring the relationships between surface patches. An important part of solving this problem is determining the physical causes of the image edges separating adjacent image regions. We currently consider the simplified case of a scene illuminated by a single spectral power distribution, with no image edges caused by illumination discontinuities.

As a first step towards general physical segmentation,

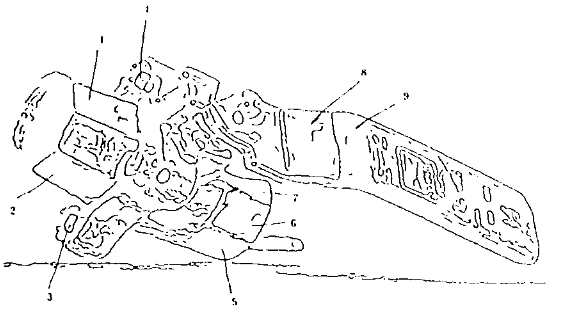


Figure 9. An edge image with labeled regions

we have developed a procedure which locates regions corresponding to surface patches of approximately the same spectral reflectance. The procedure consists of computing the normalized physical color (NPC) [7] of each image region and finding those regions which are sufficiently close in NPC space according to our color metric [7]. Given our assumption of a single spectral power distribution of illumination, regions which have similar normalized physical colors will have similar spectral reflectance functions. If our scene contains a small number of objects, it is likely that surface patches with similar spectral reflectance functions are made of the same material. Also, unless there is evidence to the contrary, adjacent image regions whose surface patches have similar spectral reflectance functions probably correspond to surface patches on a single object which are separated by a geometrical discontinuity.

6-2. Results

We have tested our procedure on an image of the valve part. Figure 9 shows an edge image obtained using Canny's edge detector [Canny, 1986]. In Figure 9 we consider nine regions which are similar to those found by the geometrical segmentation algorithm. Figure 10 shows the coordinates of these regions in NPC space. The cluster formed by regions 1,2,5,6,7 in Figure 10 indicates that the corresponding surface patches have approximately the same spectral reflectance. These regions, in fact, correspond to surface patches of the same material (the brass material on the valve). The cluster formed by regions 3,4,8 indicates that their surface patches have nearly the same spectral reflectance. These regions, in fact, correspond to silver colored patches on the valve. Region 10, the yellow handle, maps to a separate part of NPC space. Thus, we see that this procedure is very useful for grouping together surface patches which are composed of the same material. This information can readily be applied to object recognition.

It is important to emphasize that there are large differences in the image irradiance values corresponding to the regions 1,2,5,6,7 and also to the regions 3,4,8. This is because of differences in the orientation of the surface patches. Therefore, a clustering technique based on image irradiance would not be able to group these sets of regions. It is only by using normalized color that we are able to group together regions corresponding to surface patches made of the same material.

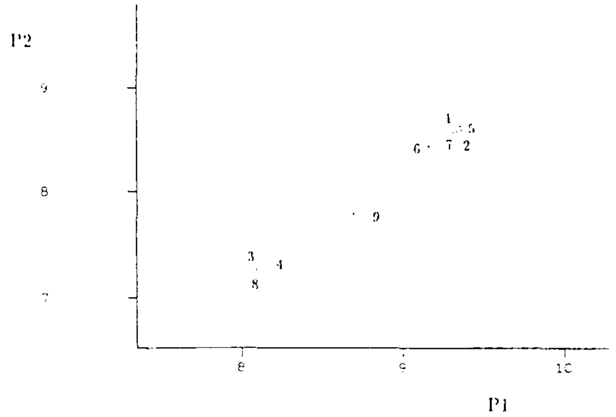


Figure 10. Region coordinates in NPC space

References

1. Ayache, N., and Faugeras, O., "HYPER: A New Approach for the Recognition and Positioning of two-dimensional objects", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. PAMI-8, pp. 44-54, 1986.

2. Ballard, D.H., "Strip Trees: a Hierarchical Representation for Curves", Comm. of the ACM, Vol. 24, No. 5, pp. 310-321, May 1981.

3. Binford, T.O, "Visual Perception by Computer", Proc. IEEE Conf. on Systems and Control, Miami, Dec. 1971.

4. Binford, T. O., Levitt, T. S. and Mann, W. B. "Bayesian Inference in Model-Based Machine Vision", Proceedings of the Workshop on Uncertainity in Artificial Intelligence, 1987.

5. Canny, J. "A Computational Approach to Edge Detection", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. PAMI-8, No. 6, pp. 679-697, November 1986.

6. Grimson, W.E.L., and Lozano-Perez, T., "Localizing Overlapping Parts by Searching the Interpretation Tree", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. PAMI-9, No. 4, pp. 469-482, 1987.

7. Healey, G. and Binford, T.O., "A Color Metric for Computer Vision," Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, June 1988. Also appears in Proceedings of ARPA Image Understanding Workshop, April, 1988.

8. Healey, G. and Binford, T.O., "Predicting Material Classes," Proceedings of ARPA Image Understanding Workshop, April, 1988.

9. Healey, G. and Binford, T.O., "Local Shape from Specularity," *Computer Vision, Graphics, and Image Processing*, April, 1988.

10. Healey, G. and Binford, T. O., "The Role and Use of Color in a General Vision system", Proceedings of the ARPA Image Understanding Workshop, USC, 1987.

11. Kanade, T., "Recovery of the 3-D Shape of an Object from a Single View", Artificial Intelligence 17, pp. 409-460, 1981.

12. Kong, S. Y. "Color segmentation", MSAI Report, Stanford University, May 1985.

13. Lowe, D., "The Viewpoint Consistency Constraint", International Journal of Computer Vision, Vol. 1, No. 1, pp. 57-72, 1987.

14. Minsky, M. L., and Papert, S. *Perceptrons; an Introduction to Computational Geometry*, Cambridge, Mass.: MIT Press, 1969.

15. Nalwa, V.S., and Binford, T.O., "On detecting edges" IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. PAMI-8, No. 6, pp. 699-714, November 1986.

16. Nalwa, V. S., Pauchon, E. "Edgel-aggregation and Edge-description", Computer Vision, Graphics and Image Processing, 40:79-94, 1987.

17. Nevatia, R. and Binford, T. O. "Description and Recognition of Curved Objects", Artificial Intelligence 8 (1977), pp 77-98.

18. Ohta, Y. it Knowledge-based Interpretation of Outdoor Natural Color Scenes, Pitman advanced publishing program, 1985.

19. Ponce, J., "Ribbons, Symmetries, and Skewed Symmetries", Proc. Image Understanding Workshop, February 1988.

20. Ponce, J., Chelberg, D. M., and Mann, W. B. "Analytical Properties of Generalized Cylinders and Their Projections", Proc. Image Understanding Workshop, February 1987.

21. Shafer, S., "Using Color to Separate Reflection Components," University of Rochester TR 136, April 1984.

ADAPTIVE SMOOTHING FOR FEATURE EXTRACTION[1]

Philippe Saint-Marc and Gérard Medioni

Institute for Robotics and Intelligent Systems
University of Southern California
Los Angeles, California 90089-0273

ABSTRACT

We present a method to smooth a signal - whether be it an intensity image, a range image or a contour - which preserves discontinuities and facilitates their detection. This is achieved by repeatedly convolving the image with a very small averaging filter modulated by a measure of the "continuousness" at each pixel. This process is related to the Anisotropic Diffusion reported by Perona and Malik recently, but is not subject to instability or divergence. We first show how this basic approach can be applied when it is reasonable to model an image as piecewise constant (such as most aerial our outdoor images), then turn to the case when it becomes necessary to detect discontinuities of the first derivatives, as is the case for range images, and finally present the case when the input signal is a connected curve. The method is extremely attractive as there is a single parameter to adjust, analogous to the variance σ in the Scale-Space paradigm, and produces a robust segmentation as no tracking is needed. We illustrate the method with results on real intensity and range images, and on actual contours.

1 Introduction

In order to successfully accomplish the understanding of images, it is necessary to transform the viewer-centered input into object-centered descriptions. Physical boundaries of objects are very important descriptors and are likely to generate edges during the imaging process. Even though the reverse is not true, it is reasonable to assume that the early stages in image analysis consist of detecting such discontinuities. Due to the complexity of the physical world and of the imaging apparatus, and to multiple sources of noise, the signal to be processed is complex, and the detection of such discontinuities is non trivial. Features detected locally are validated only by considering a more global context. We begin by reviewing the major approaches chosen to tackle this challenging problem and classify them in three categories:

- *Fixed Mask methods*, in which the model consists of a single edge, and the coefficients of the convolution mask are evaluated to best detect such an edge. An example of such a scheme is gaussian smoothing. The main drawback of these methods is that, since they do not explicitely model multiple edges in the smoothing window, edges are displaced or not detected if they occur too close together.
- *Scale-Space methods* propose to look at the signal after smoothing with multiple size masks. Since they use multiple scales, they need to either solve a non trivial correspondence problem or to compute a very large number of close scales.
- *Adaptive methods* either change the coefficients or the size of the masks based on the values in a window. Some of them are computationally expensive or use a rather adhoc strategy, but recently Perona and Malik [24] presented an interesting formulation of the problem.

Following the survey, we present a simpler variation of this last approach, in which we iteratively convolve the image with a mask whose coefficients reflect the degree of continuity of the underlying image surface. This Adaptive Filter is applicable when it is reasonable to assume that the image is piecewise constant. We then study images in which this assumption does not hold and where we have to detect roof edges in addition to step edges, such as range images. We also present a short extension of the method to 1-D signals such as the bounding contour of an object. In all cases, we show that the desired features are enhanced and their detection becomes very easy, and illustrate our claims with results on real signals. Finally, we discuss the limitations of the approach and propose some possible improvements and extensions.

[1] This research was supported by the Defense Advanced Research Projects Agency under contracts DACA76-85-C-0009 and F33615-87-C-1436, and monitored by the U.S. Army Topographic Laboratories.

2 Previous Work

As mentioned in the introduction, it is convenient to classify the Feature Extraction methods in three classes, even though the boundaries between them may not be sharp.

2.1 Fixed Mask Methods

In this group, the authors consider the case of a single edge. The goal of their methods is to find the optimal filter (in terms of signal to noise ratio) for the detection of such an edge. We only report the major ideas in this domain. Shanmugam *et al* [29] define an edge as a step discontinuity between regions of uniform intensity and show that the ideal filter is given by a prolate spheroical wave function. Marr and Hildreth [19], extending the work of Marr and Poggio [18], convolve the signal with a rotationally symmetric Laplacian of Gaussian mask and locate zero-crossings of the resulting signal. In their work, they already mention that a multiple scale approach is necessary, pointing out the difficult problem of integration. Haralick [14] locates edges at the zero-crossings of the second directional derivative in the direction of the gradient where derivatives are computed by interpolating the data. In [13], his facet model is extended to the Topographic Primal Sketch. In a recent paper, Shen and Castan [30] propose an optimal linear filter in which the image is convolved with the smoothing function $f(x) = -\frac{1}{2}(\ln b)b^{|x|}$ prior differentiation. They claim better localization than the Marr-Hildreth zero-crossing detector.

Since these methods ignore the occurence of multiple interferring edges, they typically displace the true location of edges, or worse, fail to detect some of them. When the problem is displacement only, a nice method can recover the true location of the edges [7], but still leaves the no-detection problem open.

2.2 Scale-Space Methods

As noted by several authors in the first group, the automatic adjustement of the size (or scale) parameter is difficult, so using multiple scales should provide a reasonable answer. This idea is based on some physiological work echoed in [17] for a few scales, but the integration of these discrete scales is an open problem. Instead of using discrete listant scales, Witkin [33] proposed a continuum of scales and showed that, at least in 1-D, the interpretation of the multiscale response made the important information explicit. In the 2-D case, the discretization of the formulation leads to large amount of memory allocation necessary, such as in edge focusing [3], otherwise heuristics need to be applied to establish correspondence between scales. This was done with some success by Asada and Brady for 2-D curves [1] in their Curvature Primal Sketch, then by Ponce and Brady [25] and Fan *et al* [10] in the case of surfaces. In his paper [6], Canny defines a set of criteria for the integration of multiple size masks, but his implementation is with fixed size masks only.

2.3 Adaptive Methods

The general idea behind adaptive smoothing is to evaluate the best smoothing mask in function of the available information in the signal to be smoothed. Even though detailed overviews of some Adaptive Smoothing methods can be found in [8,20] in which some evaluations are also performed, it is interesting to recall some ideas which form the basis of many of these methods, including the newest approaches.

One of the first interesting investigation in this field may be found in [16] in which Rosenfeld *et al* propose some iterative weighted averaging methods. In particular, they propose to apply at each point a weighted mask whose coefficients are based on an evaluation of the difference between the value at the center point and the values of its neighbors. A similar approach, but simpler, can be found in [32], in which the weighting coefficients are the normalized gradient-inverse between the current point and each neighbor. Another method consists in selecting the neighbor points which have the closest value from the value of the central point and replace the latter by the average of these values [9]. More sophisticated methods are based upon local statistic studies of each point neighborhood [20]. The major drawback of these iterated smoothing methods is that their convergence capability toward a stable state is not clear. In more recent papers [5,25], Brady *et al* prevent smoothing across previously detected discontinuities by using computational molecules proposed by Terzopoulos [31]. This implies that they already have detected discontinuities, which is the problem we have to solve! Geman and Geman [12] appeal to the simulated annealing which is computationally very expensive but leads to impressive results which are shown only on gray scale images with a few gray levels. A completely different approach which makes use of the curvature of the underlying image surface is proposed by Saint-Marc and Richetin [28] in which they apply a directional mask in the direction of least curvature in highly curved areas or a standard averaging square mask otherwise. The results are impressive, but this method is only applicable to surfaces smoothing, not to planar curves for instance. Parvin and Medioni [23] present a method to extract meaningful features from range images. Their strategy consists of selecting automatically an adequate kernel size for the detection and localization of such features. Although the method requires the non obvious setting of five parameters, it provides directly features at different scales and is applicable for curves. Finally, Perona and Malik [24] have proposed to cast the problem in terms of the heat equation [15], but in an anisotropic medium. They have presented impressive results on complex images. The method presented here is a refinement and a generalization of the same idea.

3 Adaptive Smoothing

In this section, we introduce our Adaptive Filter. Its performance is illustrated on a 1-D signal, then on an intensity image, both of them approximatively piecewise constant. We point out its limitations and offer solutions allowing us to deal with more complex signals.

3.1 Principle

By far the most common filter used in smoothing is the Gaussian filter. As pointed out by many authors, such a filter has very desirable properties, in particular no new zero-crossings appear in the Laplacian of the signal as σ increases [34]. In addition, Gaussian convolution can be computed efficiently by a cascade of convolutions with any finite filter [4,5,6], in particular a small mask with all coefficients set to 1. In the case of an image, we formulate this process as follows in order to later introduce our Adaptive Filter more easily. Let the image be expressed as $I(x,y,0)$ before smoothing, and let $C(x,y)$ be a coefficient image of same size such that $\forall x$ and $\forall y$, $C(x,y) = 1$, the filtered image $I(x,y,n+1)$ at the $(n+1)^{th}$ iteration is simply:

$$I(x,y,n+1) = \frac{1}{N}\sum_{k=-1}^{+1}\sum_{l=-1}^{+1} I(x+k,y+l,n)C(x+k,y+l)$$

with

$$N = \sum_{k=-1}^{+1}\sum_{l=-1}^{+1} C(x+k,y+l)$$

But it is well known that this filter smooths the data everywhere, even across the depth discontinuities. If we assume that we already know the location of these discontinuities, then if we set the corresponding points of the coefficient image $C(x,y)$ to 0, smoothing a point near a discontinuity will not take into account those points belonging to the discontinuity. As we use a small mask, there will be no risk to merge points belonging to different regions. For those points belonging to discontinuities, the repeated averaging process will force them to belong to one of the nearby regions, therefore enhancing the discontinuities. Unfortunately, we do not know the location of the discontinuities, otherwise our problem would be solved and would not need any smoothing... Instead, we can formulate a guess by computing at each point a *continuity value* using as in [24] any monotically decreasing function C such that $C(0) = 1$ and $C(d) \rightarrow 0$ as d increases, where the variable d represents the "chance" of having a discontinuity at that point. An estimate of d can be computed simply by relating its value to the value of the gradient at that point (if we assume that the original signal is approximatively piecewise constant) or it can be more elaborated as suggested in [24].

Let us now formulate our Adaptive Smoothing scheme. Let the image be expressed as $I(x,y,0)$ before smoothing, and $I(x,y,n)$ at the n^{th} iteration. From $I(x,y,n)$, we can determine the coefficient image $C(x,y,n)$ by computing at each point the *continuity value* using the previously defined function C. The smoothed version of $I(x,y,n)$ is then defined at each point (x,y) by:

$$I(x,y,n+1) = \frac{1}{R}\sum_{k=-1}^{+1}\sum_{l=-1}^{+1} I(x+k,y+l,n)C(x+k,y+l,n)$$

with

$$R = \sum_{k=-1}^{+1}\sum_{l=-1}^{+1} C(x+k,y+l,n)$$

Note that this formulation is considerably simpler than the one given in [24]. Our experiments with their original formula have led to numerical instabilities and divergence, even for simple signals. The new equation, as we will see, consistently gives stable results. So far, we have been unable to formally establish the equivalence between both formulations.

3.2 Experiments

3.2.1 1-D Signal Adaptive Smoothing

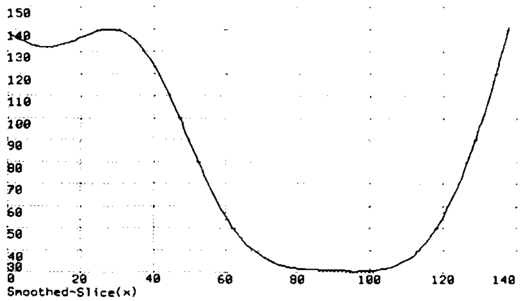
We illustrate our algorithm on a 1-D signal which consists of a slice taken horizontally from an approximatively piecewise constant gray level image shown in figure 1(a), as shown in figure 1(b). As one can notice, this noisy signal contains discontinuities of different strengths. We simply take for the estimate of d the absolute value of the derivative. We use for C the same function as in [24], that is:

$$C(d) = e^{(-k.d^2)}$$

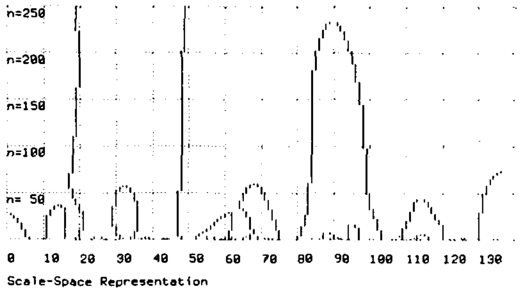
The result of the application of our Adaptive Filter to this signal after 250 iterations can be seen in figure 1(c) with the value of k set to 0.1. Notice how smooth it is and how well the discontinuities are preserved. As expected, the resulting signal is a piecewise constant function. Figure 1(f) illustrates the behavior of the zero-crossings of the second derivative of the signal as the iteration progresses.

This Scale-Space like representation really demonstrates that features detected at a coarser level (that is after a certain amount of smoothing) do not need to be tracked along the "scale" dimension in order to find their precise location as in the case of the standard Scale-Space paradigm. For comparison, figure 1(c) shows the results obtained with $k = 0$, which is identical to iteratively applying a mean filter, which is also equivalent to convolving the signal with a Gaussian Filter. Not surprisingly, discontinuities completely disappear after a number of iterations. The diagram in figure 1(d) shows how zero-crossings migrate and merge as the iteration progresses with $k = 0$, which is equivalent to increasing the σ parameter of a Gaussian convolution.

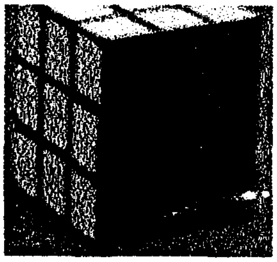
The method depends on a single parameter k, and its value is critical, as it directly determines which discontinuities will remain during the iterative process. If k is taken to be small, then every edge will diffuse, and the result will be the same as with Gaussian smoothing. If k is taken to be large, then every edge will stop the diffusion, and no smoothing will happen. Choosing the value of k is equivalent to choosing the value of σ in the standard Scale-Space paradigm from which tracking will be performed. But, in our case, no tracking along the "scale" dimension is needed.



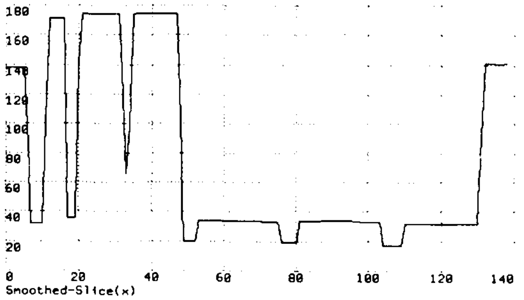
(c) Gaussian Smoothing of (b)



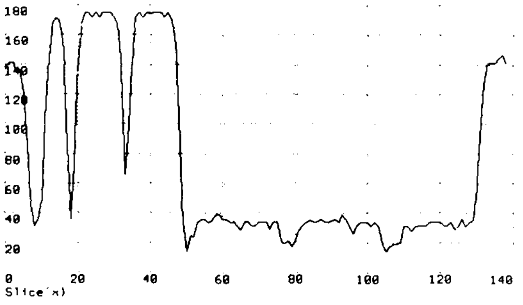
(d) Zero-Crossings at each iteration (Gaussian)



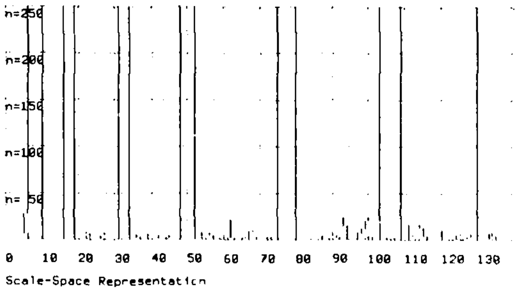
(a) Intensity Image



(e) Adaptive Smoothing of (b)



(b) 1-D Signal extracted from (a)



(f) Zero-Crossings of (b) at each iteration (Adaptive)

Figure 1: Adaptive Smoothing of a 1-D Signal.

3.2.2 Intensity Image Adaptive Smoothing

When the signal is a 2-D image, we define d as the norm of the gradient $(\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y})^{\top} = (G_x, G_y)^{\top}$, computed in a 3×3 window. Therefore, the continuity value C is given by the following equations:

$$C(d) = e^{(-k.d^2)}$$

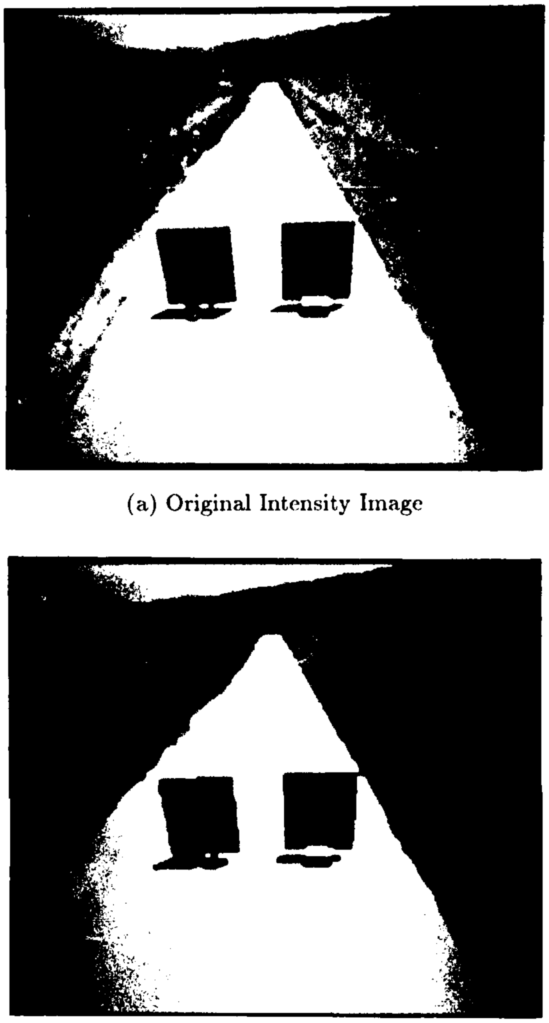
with

$$d = \sqrt{G_x^2 + G_y^2}$$

We present the results of the Adaptive Filter to the image in Figure 2(a), which is a typical scene in the environment of DARPA's Autonomous Land Vehicle. Figure 2(b) shows the result obtained after 50 iterations with the parameter k set to 0.01. Not surprisingly, the results are similar to the ones obtained for the 1-D signal. The edges are very well preserved while the image has been smoothed within regions. These results are comparable to those shown by Perona and Malik in their paper [24], but we have not been able to replicate their results using their equations.

3.3 Limitations of the method and Solutions

What happens if the original signal cannot be considered as approximatively piecewise constant? This question is very important if we wish to apply our Adaptive Filter to the smoothing of range images for instance. Furthermore, step discontinuities are not the only discontinuities that need to be preserved. In range image, one needs to preserve other types of discontinuities, such as creases. Figure 3(a) shows the $126 \times 131 \times 8$ bits range image of a toy chair obtained in our laboratory, to which Gaussian noise ($\sigma = 10$) has been added. Here, depth is directly encoded by gray-level. A better representation is a 3-D plot of the scene, as shown in figure 3(b). From this image, we extract a vertical slice displayed in figure 3(c). We apply to this 1-D signal the same algorithm as in 3.2.1, leading after 250 iterations, to the smoothed signal shown in figure 3(d). The step discontinuities are obviously still preserved, but as the signal is not piecewise constant, areas with significant values of the derivative have been transformed into staircase type signals, which is quite disappointing. Furthermore, tangent orientation discontinuities are not preserved.

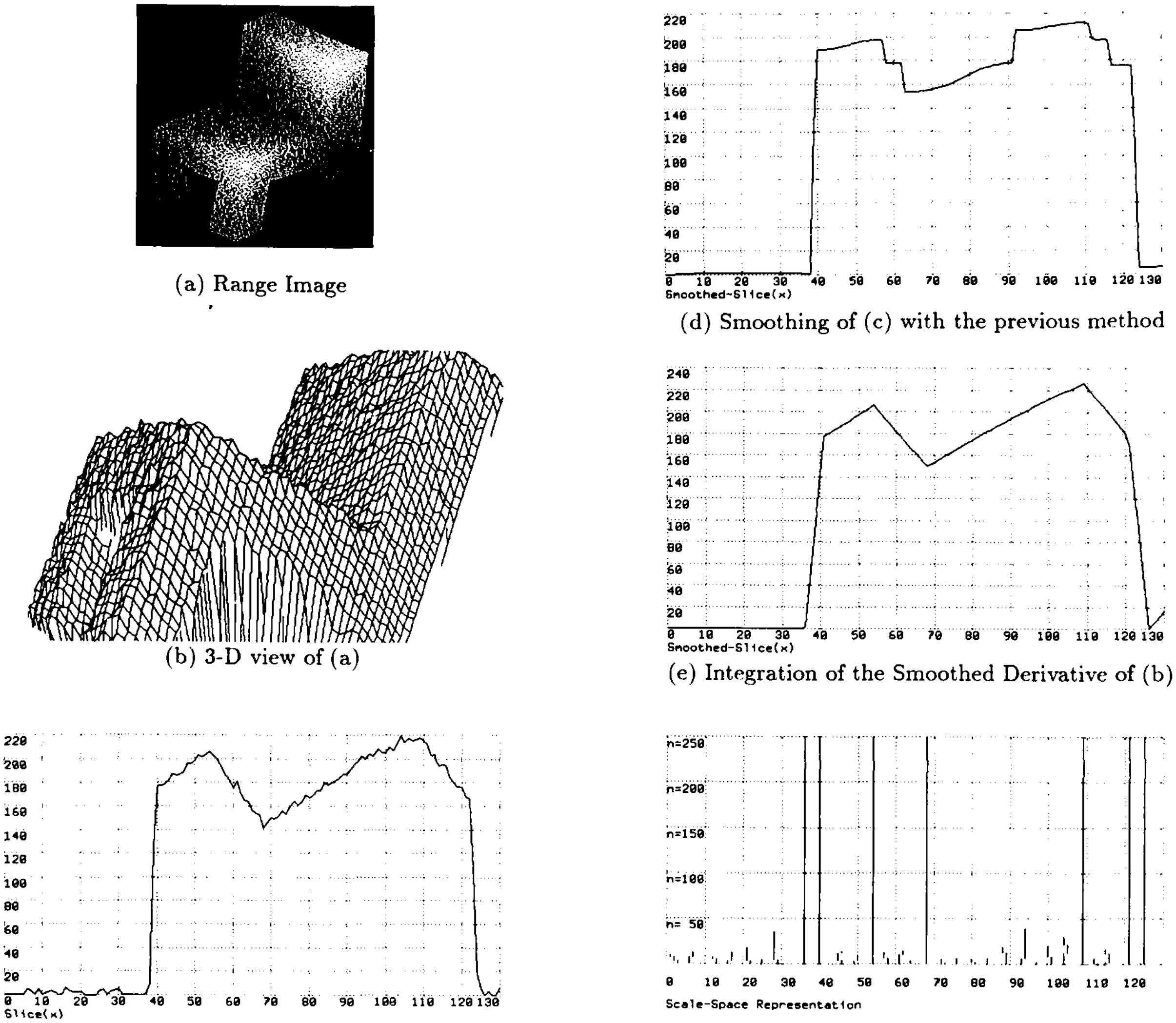


(a) Original Intensity Image

(b) Adaptive Smoothing of (a)

Figure 2: Adaptive Smoothing of an Intensity Image.

There is an elegant solution that solves both problems at the same time: instead of applying the Adaptive Filter to the signal itself, we apply it to its derivative. Figure 3(e) shows the result obtained with this method. Since we filter the derivative only, the smoothed signal is not directly accessible. For the purpose of display only, we have performed a numerical integration of the smoothed derivative. The noise has been removed while discontinuities, including tangent discontinuities, have been preserved. Finally, figure 3(f) shows in Scale-Space like representation the location of the extrema of the second derivative as the iteration progresses. These features which correspond to discontinuities of the first derivative are very well preserved along the "scale" dimension.



(a) Range Image

(d) Smoothing of (c) with the previous method

(b) 3-D view of (a)

(e) Integration of the Smoothed Derivative of (b)

(c) 1-D Signal extracted from (a)

(f) Extrema of the Second Derivative at each iteration

Figure 3: Preserving First Derivative Discontinuities.

4 Application to Range Image Segmentation

In this section, we show how our Adaptive Filter previously introduced can be extended to the smoothing of range data. As this smoothing scheme preserves important surface changes, they are then very well extracted by standard means. Furthermore, as the resulting surfaces are smooth, low level frequency events are also detectable.

4.1 Range Image Adaptive Smoothing

The smoothing is performed as follows. Given the original range image $R(x,y,0)$, we first compute the original derivative images $P(x,y,0) = \frac{\partial R(x,y,0)}{\partial x}$ and $Q(x,y,0) = \frac{\partial R(x,y,0)}{\partial y}$. Let $P(x,y,n)$ and $Q(x,y,n)$ the derivative images at the n^{th} iteration. From these two images we compute the coefficient image $C(x,y,n)$ using at each point the following formula:

$$C(d) = e^{(-k.d^2)}$$

with

$$d = \Delta = \frac{\partial P}{\partial x} + \frac{\partial Q}{\partial y}$$

(Laplacian)

The smoothed versions of $P(x, y, n)$ and $Q(x, y, n)$ are then obtained by using the coefficient image $C(x, y, n)$ previously defined, as follows:

$$P(x,y,n+1) = \frac{1}{S} \sum_{k=-1}^{+1} \sum_{l=-1}^{+1} P(x+k,y+l,n)C(x+k,y+l,n)$$

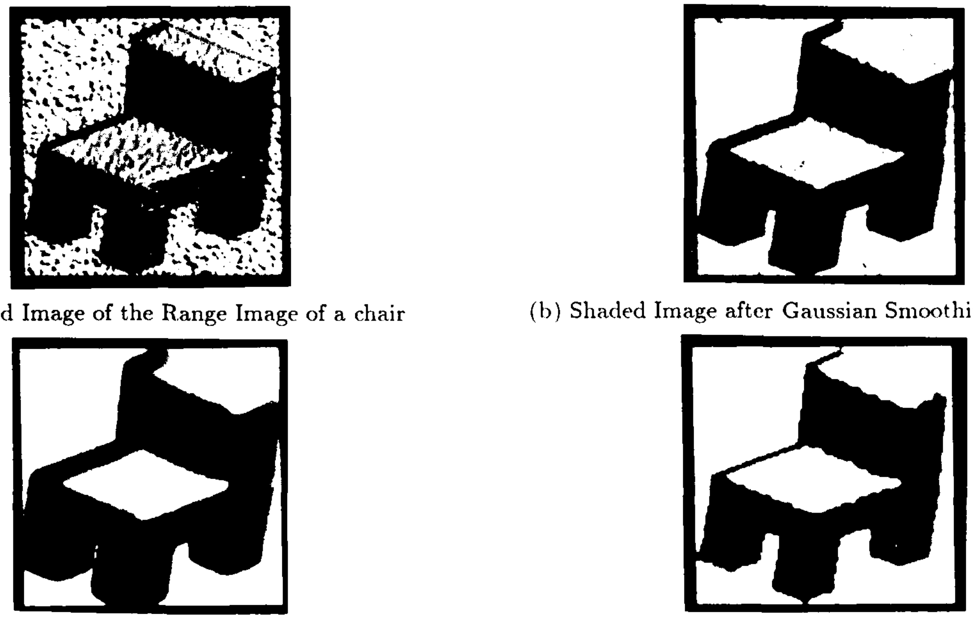
$$Q(x,y,n+1) = \frac{1}{S} \sum_{k=-1}^{+1} \sum_{l=-1}^{+1} Q(x+k,y+l,n)C(x+k,y+l,n)$$

with

$$S = \sum_{k=-1}^{+1} \sum_{l=-1}^{+1} C(x+k, y+l, n)$$

It is important to notice that $C(x, y, n)$ *simultaneously* reflects the continuity of both $P(x, y, n)$ and $Q(x, y, n)$, so that the smoothing of $P(x, y, n)$ and $Q(x, y, n)$ cannot be performed independently, which is legitimate. The choice of the Laplacian is not unique, but is the simplest. It would have been possible for instance to consider the quadratic variation or another measure of the local variation of the first derivatives.

We have tested the algorithm on several range images with success. In order to visualize the performance of the Adaptive Filter applied to a range image, a possible solution is to integrate the smoothed derivative images P and Q in order to obtain a smoothed range image and then take a 3-D view. Instead, we found more practical to directly use P and Q and compute with these the shaded image of the smoothed range image. Figure 4 shows several results obtained with the range image of figure 3(a) whose shaded image is shown in figure 4(a). This range image includes step discontinuities corresponding to transitions between the object and the background, and also roof discontinuities corresponding to transitions between planes. Figure 4(d) shows the result obtained after only 10 iterations of our algorithm, in which $k = 0.1$. As one can see, the image is almost perfectly smooth, without any blurring around the discontinuities. As a comparison, we show the results obtained after simply applying Gaussian convolution to the range image with $\sigma = 2$ (figure 4(b)) then with $\sigma = 4$ (figure 4(c)). Even with $\sigma = 4$, the noise is not completely removed (see the background) and in both cases, the "edges" are very much blurred. For completeness, we have also applied our Adaptive Filter to the smoothing of the $126 \times 160 \times 8$ bits range image of a more complex object (a tooth) whose shaded image is displayed in Figure 5(a). Besides the presence of step and roof discontinuities, this



(a) Shaded Image of the Range Image of a chair

(b) Shaded Image after Gaussian Smoothing ($\sigma = 2$)

(c) Shaded Image after Gaussian Smoothing ($\sigma = 4$)

(d) Shaded Image after Adaptive Smoothing of its Derivatives

Figure 4: Adaptive Smoothing of the Range Image of a chair.

(a) Shaded Image of the Range Image of a tooth (b) Shaded Image after Adaptive Smoothing of its Derivatives

Figure 5: Adaptive Smoothing of the Range Image of a tooth.

image is of particular interest since its surface is curved. The result after ten iterations with $k = 0.1$ is shown in figure 5(b). The discontinuities are very well preserved while the curved surface is now completely smoothed as the irregularities of the original surface have completely disappeared.

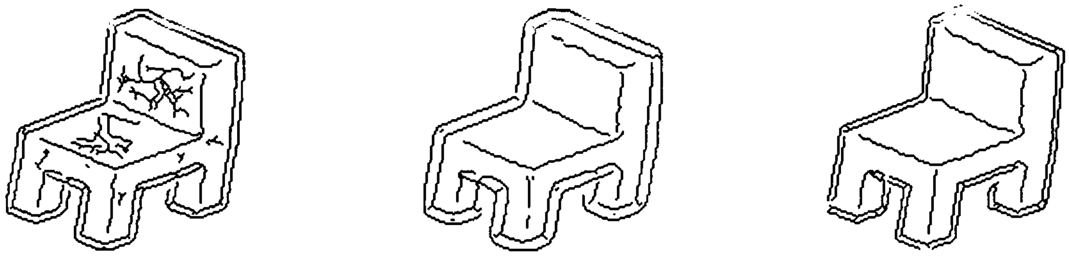
4.2 Extraction of Meaningful Features

The pictures presented in the previous section show that the algorithm produces a striking *visual* improvement, we now show how it also facilitates the feature extraction task. Surfaces curvature has been widely used recently by many researchers to achieve range image segmentation [2,5,10,25]. Indeed, local curvature is an excellent tool to characterize a surface, especially because it provides *intrinsic* properties of the surface. The extraction of meaningful features from range images is thus possible by locally observing the behavior of the curvature. Zero-crossings and extrema of the maximal curvature are particularly interesting. Unfortunately, as curvature is very sensitive to noise, it is often difficult to both detect and localize *precisely* such events. Ponce and Brady [25] and Fan *et al* both propose to solve this problem by integrating multiple scales, in effect *detecting* features at a coarser scale and *locating* then at a fine scale. Since there is, in general, no one-to-one correspondence, they use heuristics to resolve ambiguities.

We now demonstrate how our Adaptive Filter greatly facilitates the extraction of meaningful features from range images without the drawbacks of the other methods. As this smoothing scheme preserves surface discontinuities, their detection **and** their localization is immediate. Furthermore, as this smoothing process is applied to the first partial derivatives, they are directly available to compute the second partial derivatives and hence the principal curvatures. Figure 6 shows the extraction of the local extrema of the maximal curvature from the range image of the chair. This extraction is performed as follows: we fisrt label all the points in the image which correspond to extrema of the maximal curvature (in the direction of the maximal curvature), then starting from all points at which the maximal curvature is above a certain threshold τ, we extend the "contours" using hysteresis [6]. The same threshold was used for the three images. Figure 6(a) and figure 6(b) show the results obtained respectively with the (gaussian) blurred range images of figure 4(b) ($\sigma = 2$) and figure 4(c) ($\sigma = 4$). The first one is not smoothed enough, therefore "false" extrema are extracted. The second one is smoothed enough but their is a lot of distorsion which causes the extrema to be delocalized. Figure 6(c) shows the result obtained by using our Adaptive Filter: the extrema are very well detected **and** localized. It is also important to note that our method is very insensitive to the choice of the threshold τ. For completeness, figure 7 shows the extraction of the zero-crossings and extrema of the maximal curvature from the range image of the tooth. This image is more complex with slow variations of the curvature. The results prove that at the same time low frequency events and high frequency events (discontinuities) are detected without any cost in the localization of the discontinuities. Again, this method does not require any complex tracking along different scales, or any previous knowledge about discontinuities. Our Adaptive Filter is now currently used in our laboratory by our 3-D Object Recognition System for the segmentation of the range images [11], replacing the heuristic steps described above.

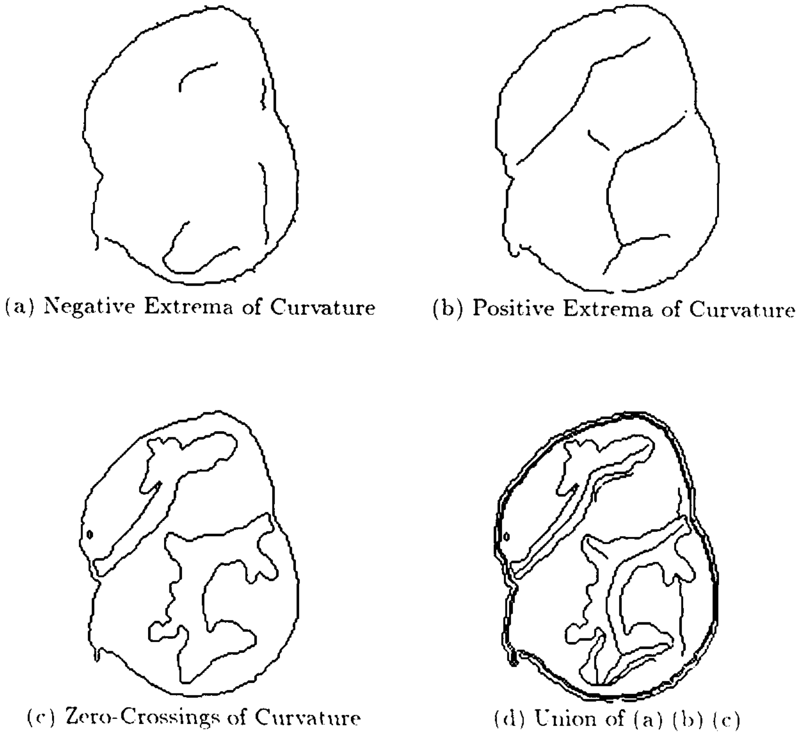
5 Application to Planar Curve Corner Detection

In this final section, we show how our Adaptive Filtering method can be applied to the smoothing of curves (here planar) in order to extract meaningful features such as corners in the bounding contour of an object.



(a) After Gaussian Smoothing ($\sigma = 2$) (b) After Gaussian Smoothing ($\sigma = 4$) (c) After Adaptive Smoothing

Figure 6: Extrema of Curvature of the Range Image of a chair



(a) Negative Extrema of Curvature (b) Positive Extrema of Curvature

(c) Zero-Crossings of Curvature (d) Union of (a) (b) (c)

Figure 7: Curvature Features of the Range Image of a tooth

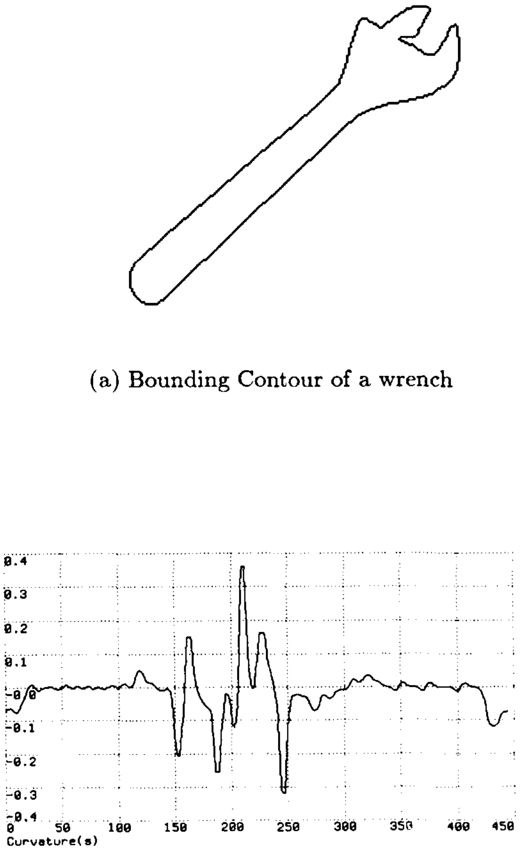
A planar curve such as the bounding contour of an object is completely defined by the parametrization $(x(s), y(s))$, where $x(s)$ and $y(s)$ are two continuous functions of the variable s. Usually, these two functions are not smooth, because of the discretization of the bounding contour. Therefore, smoothing is generally necessary when significant local events need to be extracted from this contour. Such meaningful features can consist of discontinuities of the tangent (corners), discontinuities of the curvature (smooth join) or inflection points along the curve [1,22,21]. As our Adaptive Filter smooths any original signal while preserving its discontinuities, it seems appropriate to use it for such a purpose.

If we wish to extract corners, the first idea that comes to mind is to take as original signal the tangent along the curve $\theta(s)$ where:

$$\theta(s) = \arctan \frac{y(s)}{x(s)}$$

Thus, the Adaptive Smoothing of this 1-D signal would be performed as in section 3.2.1. But we have seen that if such a signal is not approximatively piecewise constant, false discontinuities are created (see section 3.3.3). Now, the function $\theta(s)$ is rarely piecewise constant, except for polygonal contours. The solution is, as proposed in section 3.3.3, to smooth $\theta'(s) = \frac{d\theta(s)}{ds}$. Thus the Adaptive Filter applied to this signal will smooth it while preserving its discontinuities, which are identical to curvature discontinuities.

Figure 8(a) shows the bounding contour of a wrench and figure 8(b) the corresponding 1-D signal $\theta'(s)$ which we call curvature (the 1-D signal corresponding to the actual curvature is very similar). As one can see, there are fluctuations of the curvature along the curve which are not meaningful (the 1-D signal starts at the far left point of the planar curve and follows the contour clockwise). The result of the application of the Adaptive Filter to this signal can be seen in figure 10(a) after 100 iterations. Curvature discontinuities are enhanced while the signal has been smoothed between them. It is then possible to extract all curvature discontinuities (smooth joins) and associate those corresponding to the same event (corners). Another possibility is to use that 1-D signal to encode the curve into piecewise constant areas. Here, for the sake of demonstration, we only extract the extrema of the curvature. The diagram of figure 10(b) shows the behavior of these peaks of curvature as the iteration progresses. Again, there is neither displacement nor merging of the features. Figure 10(c) shows the corresponding points superimposed on the contour. The filled circles correspond to positive extrema of the curvature and the others to negative extrema. This really illustrate how well the corners are detected **and** localized by using the Adaptive Filter. For comparison, we have smoothed the signal of figure 8(b) by repeated averaging with $k = 0$ which gives after 100 iterations the signal of figure 9(a). This signal is very much blurred and the diagram of figure 9(b) shows how the extrema of the curvature are displaced as iteration progresses, some of them being "lost" after a certain amount of smoothing as nearby features have merged. Figure 9(c) shows the corresponding points on the curve.

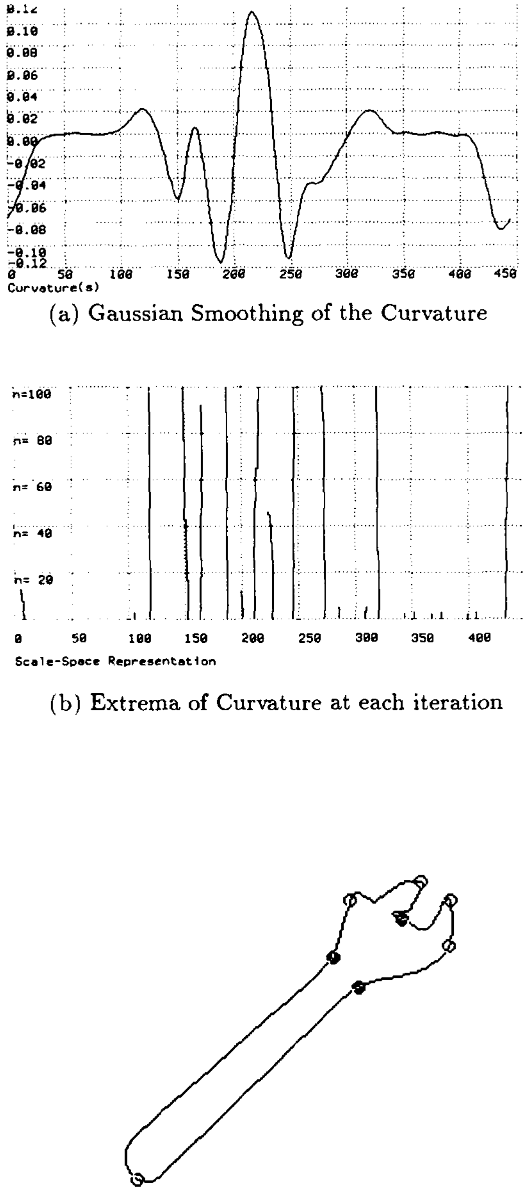


(a) Bounding Contour of a wrench

(b) Curvature along the 2-D curve

Figure 8:

Curvature along the bounding contour of a wrench

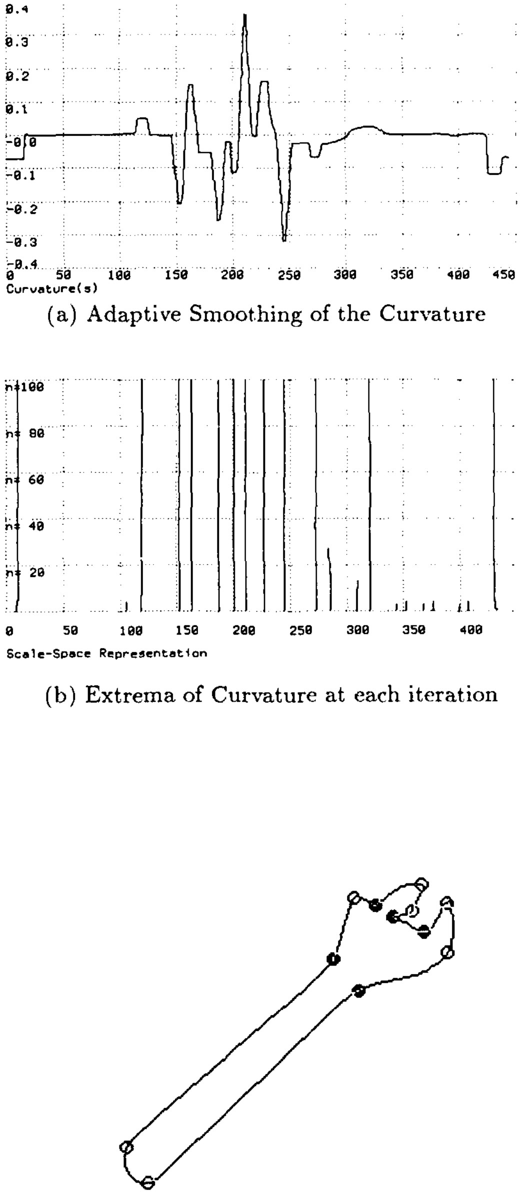


(a) Gaussian Smoothing of the Curvature

(b) Extrema of Curvature at each iteration

(c) Corner Extraction

Figure 9: Corner Extraction after Gaussian Smoothing of the Curvature



(a) Adaptive Smoothing of the Curvature

(b) Extrema of Curvature at each iteration

(c) Corner Extraction

Figure 10: Corner Extraction after Adaptive Smoothing of the Curvature

6 Conclusion

We have presented an Adaptive Filter which in its basic formulation provides a new formalism for the Anisotropic Diffusion developped by Perona and Malik. We believe that this method is simpler and give more stable results. Furthermore, we have shown that this framework can be extended easily to higher order discontinuitities preservation. Impressive results on intensity images, range images and also planar curves have been provided. Nevertheless, we think that we can improve the robustness of our Adaptive Smoothing method in particular for 2-D signals. In this case, the estimate of d can be more complex as more information such as connectivity is present in images. Also, it is important that d does not depend on the scale. We are currently investigating these ideas in order to enhance the already nice performance of our Adaptive Filter which has proven to be a powerful tool for the extraction of meaningful features.

References

[1] H. Asada and M. Brady, "The Curvature Primal Sketch," *IEEE Transactions on PAMI*, Vol. 8, January 1986, pages 2-14.

[2] P.J. Besl and R.C. Jain, "Segmentation through Symbolic Surface Descriptions," *Proceedings IEEE of Computer Vision and Pattern Recognition*, Miami, 1986, pages 77-85.

[3] F. Bergholm, "Edge Focusing," *IEEE Transactions on PAMI*, Vol. 9, November 1987, pages 726-741.

[4] J.P. Burt, "Fast Algorithms for Estimating Local Image Properties," *Journal of Computer Graphics and Image Processing*, No. 21, pages 368-382.

[5] M. Brady, J. Ponce, A. Yuille and H. Asada, "Describing Surfaces," *Journal of Computer Vision, Graphics and Image Processing*, No. 32, October 1985, pages 1-28.

[6] J.F. Canny, "A Computational Approach to Edge Detection," *IEEE Transactions on PAMI*, Vol. 8, No. 6, pages 679-698.

[7] J.S. Chen and G. Medioni, "Detection, Localization and Estimation of Edges," *Proceedings of IEEE Workshop on Computer Vision*, Miami, November 1987, pages 215-217.

[8] R.T. Chin and C.L. Yeh, "Quantitative Evaluation of Some Edge Preserving Noise Smoothing Techniques," *Journal of Computer Vision, Graphics and Image Processing*, No. 23, 1983, pages 67-91.

[9] L.S. Davis and A. Rosenfeld, "Noise Cleaning by Iterated Local Averaging," *IEEE Transactions on SMC*, Vol. 8, 1978, pages 705-710.

[10] T.J. Fan, G. Medioni and R. Nevatia, "Segmented Descriptions of 3-D Surfaces," *IEEE Journal of Robotics and Automation*, December 1987.

[11] T.J. Fan, G. Medioni and R. Nevatia, "3-D Object Recognition using Surface Description," in these Proceedings.

[12] S. Geman and D. Geman, "Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images," *IEEE Transactions on PAMI*, Vol. 6, November 1984, pages 721-741.

[13] R.M. Haralick, "Ridges and Valleys on Digital Images," *Journal of Computer Vision, Graphics and Image Processing*, No. 22, 1983, pages 28-38.

[14] R.M. Haralick, "Digital Step Edge from Zero Crossings of Second Directional Derivatives," *IEEE Transactions on PAMI*, Vol. 6, No. 1, January 1984, pages 58-68.

[15] R.A. Hummel, "Representations based on Zero-Crossings in Scale-Space," *Proceedings of IEEE Computer Vision and Pattern Recognition*, Miami, June 1986, pages 204-209.

[16] A. Lev, S.W. Zucker and A. Rosenfeld, "Iterative Enhancement of Noisy Images," *IEEE Transactions on SMC*, Vol. 7, 1977, pages 435-441.

[17] D. Marr, "Vision: A Computational Investigation into the Human Representation and Processing of Visual Information," W.H. Freeman & Co., San Fransisco, 1982.

[18] D. Marr and T. Poggio, "A Computational Theory of Human Stereo Vision," *Proceedings of the Royal Society of London*, B204, 1979, pages 301-328.

[19] D. Marr and H. Hildreth, "Theory of Edge Detection," *Proceedings of the Royal Society of London*, B207, 1980, pages 187-217.

[20] G.A. Mastin, "Adaptive Filters for Digital Image Noise Smoothing: an Evaluation," *Journal of Computer Vision, Graphics and Image Processing*, No. 31, 1985, pages 103-121.

[21] G. Medioni and Y. Yasumoto, "Corner Detection and Curve Representation using Cubic B-Splines". *Journal of Computer Vision, Graphics and Image Processing*, No. 39, 1987, pages 267-278.

[22] F. Mokhtarian and A. Mackworth, "Scaled-based Description and Recognition of Planar Curves and Two-Dimensional Shapes," *IEEE Transactions on PAMI*, Vol. 8, January 1986, pages 34-43.

[23] B. Parvin and G. Medioni, "Adaptative Multiscale Feature Extraction from Range Data," *Proceedings of IEEE Workshop on Computer Vision*, Miami, November 1987, pages 23-28.

[24] P. Perona and J. Malik, "Scale Space and Edge Detection using Anisotropic Diffusion," *Proceedings of IEEE Workshop on Computer Vision*, Miami, November 1987, pages 16-22.

[25] J. Ponce and M. Brady, "Toward a Surface Primal Sketch," *Proceedings of IEEE International Conference on Robotics and Automation*, St-Louis, March 1985, pages 420-425.

[26] W.K. Pratt, "Digital Image Processing," Wiley, New-York, 1978.

[27] A. Rosenfeld and A.C. Kak, "Digital Picture Processing," Academic Press, New-York, 1976.

[28] P. Saint-Marc and M. Richetin, "Structural Filtering from Curvature Information," *Proceedings of IEEE Computer Vision and Pattern Recognition*, Miami, June 1986, pages 338-343.

[29] F.M. Dickley and K.M. Shanmugam, "An Optimal Frequency Domain Filter for Edge Detection in Digital Pictures," *IEEE Transactions on PAMI*, Vol. 1, No. 1, pages 37-49.

[30] J. Shen and S. Castan, "An Optimal Linear Operator for Edge Detection," *Proceedings of IEEE Computer Vision and Pattern Recognition*, Miami, June 1986, pages 109-114.

[31] D. Terzopoulos, "The Role of Constraints and Discontinuities in Visible Surface Reconstruction," *Proceedings of IJCAI*, Karlsruhe, 1983, pages 1073-1077.

[32] D.C.C. Wang, A.H. Vagnucci and C.C.Li, "Gradient Inverse Weighted Smoothing Scheme and the Evaluation of its Performance," *Journal of Computer Graphics and Image Processing*, No. 15, 1981, pages 167-181.

[33] A.P. Witkin, "Scale-Space Filtering," *Proceedings of IJCAI*, Karlsruhe, 1983, pages 1019-1022.

[34] A.L. Yuille and T.A. Poggio, "Scaling Theorems For Zero-Crossings," *IEEE Transactions on PAMI*, Vol. 8, January 1986, pages 15-25.

Recognizing Solid Objects by Alignment

Daniel P. Huttenlocher
Shimon Ullman

Artificial Intelligence Laboratory
Massachusetts Institute of Technology
Cambridge, Massachusetts 02139

Abstract. This paper shows that a correspondence between three points on a rigid solid object and three points in a two-dimensional image is sufficient to *align* the object with the image. The method uses a "weak perspective" viewing model, where true perspective is approximated by orthographic projection plus scale. We show that the alignment transformation exists, and is unique up to a reflection, for any non-colinear triple of object and image points. The transformation can also be computed from edge fragments, where the endpoints are unknown. For planar objects, the transformation is equivalent to a two-dimensional affine transform from the object to the image. We have implemented a recognition system that uses one or two pairs of model and image features to hypothesize potential alignments of object models with an image. Each hypothesis is verified by projecting the model features into the image. Examples using points and edges extracted from grey-level images are shown.

1 Introduction

Object recognition is generally formulated as the problem of matching one or more object models to an image containing zero or more instances of these objects. The matching process finds instances of the models in the image, and for each instance determines a transformation mapping the model onto the instance. This transformation generally specifies the position and orientation of the object with respect to the viewer (the object pose), although for certain tasks it does not.

A number of systems have been developed within this model based recognition framework (for a recent review [1]). These systems generally address tasks where the imaging process does not involve projection from the world into the image; for instance, tasks where the objects are in a known plane (2D recognition), or tasks where the "image" is three-dimensional, specifying a distance to the viewer at each pixel (3D from 3D recognition). This restriction eliminates the problems of identifying image features under projection, and recovering the three-dimensional position and orientation of an object from a two-dimensional view.

In this paper, we address the problem of recognizing rigid solid objects with unknown position and orientation in three-space, from a single two-dimensional image. The approach is to determine possible alignments of a model with an image, and then verify each hypothesized position and orientation by transforming the model into image coordinates. The method is an extension of our earlier work on aligning objects with images [8]. There are two significant advances reported here. First, we show that the alignment transformation exists and is unique (up to a reflection) for any triple of non-colinear model points. We also present a closed form solution for the alignment transformation that is simple to compute. Second, we have implemented an alignment based recognizer for solid objects, whereas our earlier implementation was restricted to planar objects.

1.1 The Imaging Model

3D from 2D recognition involves projection from the world into the image, so the imaging process must be modeled in order to recover the position and orientation of an object with respect to an image. Throughout this paper, we assume a coordinate system with the origin on the image plane, I, and with the z-axis normal to I.

Perspective projection is the most accurate model of the imaging process. As shown in Figure 1, under this model the center of projection, f, is defined to be a point along the viewing axis, v, which is perpendicular to the image plane, I. Each point in the world, p, is connected to its image, p', in I, by a ray, P, that passes through the

Support was provided in part by the Advanced Research Projects Agency of the Department of Defense under Army contract DACA76-85-C-0010, in part by the Office of Naval Research University Research Initiative Program under Office of Naval Research contract N00014-86-K-0685, and in part by the Advanced Research Projects Agency of the Department of Defense under Office of Naval Research Contract N00014-85-K-0124.

focal point f. The focal length is the distance along v from f to the image plane, I.

While perspective projection is an accurate model of imaging, it is relatively complicated to recover the position and orientation of an object from an image under perspective projection. In general, solving for the transformation requires six model points and six corresponding image points [6]. Furthermore, the equations are relatively unstable, so in practice getting a good estimate of the transformation requires more than six pairs of points, and the use of a least squares or other error minimization procedure (such as in [12]).

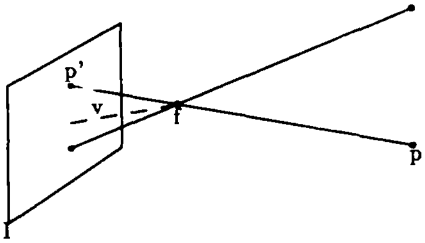


Figure 1. Perspective projection.

Under orthographic (or parallel) projection, each point, p, in the world is connected to its image, p', in the image plane, I, by a ray, P, which is perpendicular to I, as shown in Figure 2. The major difference between perspective and orthographic projection is that under perspective objects that are further away appear smaller. Thus, if a linear scale factor is added to orthographic projection, a relatively good approximation to perspective is obtained. The approximation becomes poor when the object being imaged is very deep with respect to the field of view, because a single scale factor cannot be used for the entire object [14]. For instance, railroad tracks going off towards the horizon, or an object viewed from very close-up are not well approximated by this model. Orthographic projection plus scale has been termed "weak perspective", because it approximates perspective well under most viewing conditions.

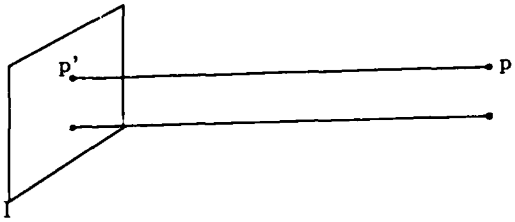


Figure 2. Orthographic projection.

Under weak perspective, all z-positions are equivalent. Thus, the transformation specifying the position and orientation of an object with respect to an image consists of a two-dimensional translation (in x and y), a three-dimensional rotation, and a scale factor that depends on the distance to and size of the object. Such a transformation, which preserves relative lengths, is called a similarity transform.

1.2 Transforming a Model to an Image

The problem of recovering a transformation that maps an object model onto an image has been a major focus of research on object recognition. Two general methods have been used for recovering the transformation. The first method computes global properties of an image (such as moments of inertia), and determines the transformation from these image properties and corresponding model properties. The problem with this approach is that the properties are based on the entire image and the entire model. Therefore, the method is not applicable to an image that contains multiple objects, or partially occluded instances of an object. The second method uses corresponding local model and image features to recover the transformation. These features are local, so occlusion and multiple objects are less of a problem. A local feature, however, does not generally contain enough information to solve for the transformation, so groups of model and image features must be used. Therefore this approach can involve substantial search among the sets of possible corresponding model and image features.

The idea behind the alignment method presented here is to identify the minimum amount of information needed to solve for a possible position and orientation, and thereby to minimize the amount of search required in matching local model and image features. In Section 2 we show that under weak perspective, three non-colinear model points and three corresponding image points are sufficent to align a model with an image. In contrast, under perspective projection it takes six corresponding model and image points to determine the position and orientation of a solid object.

The weak perspective imaging model has been used in a number of 3D from 2D recognition systems [2] [4] [11] [14]. These systems do not, however, solve the problem of computing the transformation for positioning and orienting a rigid solid object given an image. Instead, they either use a heuristic approach to recovering the transformation [2], use approximate solutions [13], store views of an object from all possible angles in order to approximate the transformation by table lookup [14], or restrict recognition to planar objects [4] [11].

1.3 Recognition by Alignment

In Section 3 we describe the ORA (Object Recognition by Alignment, pronounced "ORA") system that uses the alignment transformation to hypothesize possible positions and orientations of objects in an image. Rather than computing the transformation from triples of model and image points, the recognizer uses one or two features that define more than a single point. These features are computed from the intensity edges in a grey-level image.

There are two classes of features, based on the amount of position and orientation information that a feature specifies. Class I features specify a triple of points, and Class II features specify an oriented point. A single Class I feature in an image and a corresponding model feature is enough to compute the alignment transformation, while a pair of corresponding Class II features are required. Class I features are, for example, a vertex connected to two complete edges, or an edge connected to two partial edges. Class II features are, for example, a vertex connected to two partial edges, a curved arc, or two nearby edge fragments.

For a given model, each Class I feature is matched with each image feature of the same type, and the resulting alignment transformations are computed. A transformation is scored based on the number of model features that are brought into approximate correspondence with an image feature. Those transformations accounting for more than half of a model's features are kept as matches. Once the Class I features are exhausted, each pair of Class II model features is matched with each pair of Class II image features that do not correspond to some already aligned model.

The worst case running time of the matching process is $O(m^3 i^2)$ for m model features and i image features, because it may be necessary to consider all pairs of model features against all pairs of image features, and scoring each match involves transforming all the model features to image coordinates. In practice, however, it is not necessary to consider all pairs of features, because the transformations computed from the Class I features have already accounted for many of the Class II image features. Furthermore, to the extent that any grouping or classification of features can be done, the number of matches considered can be reduced by using only certain pairs of features rather than taking all pairs.

2 The Alignment Method

The problem of accurately and efficiently determining potential alignments of a model with an image is a central problem in recognition. In this section we show that three pairs of corresponding model and image points specify a unique (up to a reflection) position and orientation of a model with respect to an image. In the case of a planar model, the alignment transform is equivalent to an affine transform from the model plane to the image plane. Unlike the affine transform, however, the alignment transform applies to solid models.

A closed form solution for computing the alignment transform directly from three pairs of corresponding model and image points is also presented. The method also applies to two pairs of oriented model and image points, or to three pairs of model and image edge fragments (without knowing the endpoints of the edges). Finally, the issue of recovering non-rigid transformations is addressed.

2.1 The 2D Affine Transform

Some systems for 3D from 2D recognition of planar objects use a two-dimensional affine transform to map the object plane to the image plane [4] [11]. A two-dimensional affine transform, $A : \mathbf{x} \rightarrow \mathbf{x}'$, can be represented as a non-singular 2×2 matrix, $\mathbf{L}$, and a two-dimensional vector, $\mathbf{b}$, such that $\mathbf{x}' = \mathbf{L}\mathbf{x} + \mathbf{b}$. An affine transformation preserves parallelism, but captures scaling and shearing of a plane.

A two-dimensional affine transform can be used to map a plane in space onto its image under orthographic projection plus scale [11]. The converse, that a two-dimensional affine transform corresponds to a position, orientation and scale of a plane in space is not, however, an established result. Therefore, approaches that solve for an affine transform from a model plane to an image [4] [11] have not established that a solution must correspond to a possible three-dimensional position and orientation of the model plane.

The proof in the next section, that three non-colinear model and image points always uniquely define a three-dimensional alignment transformation, makes use of the two-dimensional affine transform from a model plane to an image. It is shown that a two-dimensional affine transform always corresponds to a unique three-dimensional similarity transform (up to a reflective ambiguity). Thus an affine transformation mapping a model plane to the image plane does in fact always specify a three-dimensional position and orientation.

A partial version of the existence and uniqueness of the alignment transform has been shown by Kanade and Kender [10]. They use an affine transformation between two planar patches to derive a constraint on the relative orientation of the patches. First they assume that if there is a two-dimensional affine transformation $\mathbf{x}' = \mathbf{L}\mathbf{x} + \mathbf{b}$, for $\mathbf{x}$ on P_2 and $\mathbf{x}'$ on P_1, then P_1 and P_2 are projections of planar patterns P_1' and P_2' in three-dimensions, which are related by a similarity transformation (by a three-dimensional rotation, translation and scale). Then they proceed to show that this assumption constrains the relative three-dimensional orientations of P_1' and P_2' to two possible configurations, which are reflections of one another. A restricted form of this constraint is used for recovering three-dimensional shape from skewed symmetries in a two-dimensional image.

Kanade and Kender express their assumption as

$$\mathbf{L}\mathbf{T}_2 = \mathbf{T}_1 \sigma \mathbf{R},$$

where $\mathbf{T}_i$ rotates P_i' about the x and y axes so that it is parallel to the $z = 0$ plane, $\mathbf{R}$ is rotation about the z-axis (the viewing axis), and σ is a linear scale factor. From this equation, they derive two equations relating the slant and tilt of P_1' and P_2'. These equations are claimed to always have two symmetric solutions. The proof, however, relies on $\det(\mathbf{L}) > 0$, which is not the case when there is a reflection involved in the transformation from P_2' to P_1'. Furthermore, the uniqueness of the rotation about the z-axis is not established. Thus they do not show that in general a two-dimensional affine transform is always the orthographic projection of a unique three-dimensional similarity transform.

2.2 The Alignment Transformation Exists and is Unique

The major result of this section is that the correspondence of three non-colinear points is sufficient to determine the position, three-dimensional orientation, and scale of a rigid solid object with respect to a two-dimensional image. The result is shown in several stages. First it is established that an affine transformation of the plane is uniquely defined by three pairs of non-colinear points. Then a linear transformation of the plane is shown to uniquely define a similarity transformation of space, specifying the orientation of one plane with respect to another (up to a reflection).

These two results are then combined to show that three pairs of non-colinear points define the position and orientation of one plane with respect to another (up to reflection). For a rigid object, the position and orientation of one plane of the object defines the position and orientation of the entire object. Thus, the method applies directly to rigid solid objects. In the case of planar objects, the reflective ambiguity in the transformation is not detectable. For solid objects, the ambiguity may be resolved using a point not coplanar with the three alignment points.

Lemma 1. *Given three non-colinear points $\mathbf{a}_m$, $\mathbf{b}_m$, and $\mathbf{c}_m$ in the plane, and three corresponding points $\mathbf{a}_i$, $\mathbf{b}_i$, and $\mathbf{c}_i$ in the plane, there exists a unique affine transformation, $A(\mathbf{x}) = \mathbf{L}\mathbf{x} + \mathbf{b}$, for any two-dimensional vector $\mathbf{x}$, where $\mathbf{L}$ is a linear transformation and $\mathbf{b}$ is a translation, such that $A(\mathbf{a}_m) = \mathbf{a}_i$, $A(\mathbf{b}_m) = \mathbf{b}_i$, and $A(\mathbf{c}_m) = \mathbf{c}_i$.*

This fact is relatively well known in higher geometry, and its proof is simple (see for example [5] [9]).

Definition 1. *A transformation,* $\mathbf{T}$, *is a* similarity transform *over a vector space V when*

$$\|\mathbf{T}\mathbf{v}_1\| = \|\mathbf{T}\mathbf{v}_2\| \iff \|\mathbf{v}_1\| = \|\mathbf{v}_2\|$$

$$\mathbf{T}\mathbf{v}_1 \cdot \mathbf{T}\mathbf{v}_2 = 0 \iff \mathbf{v}_1 \cdot \mathbf{v}_2 = 0$$

for any $\mathbf{v}_1$, $\mathbf{v}_2$ in V.

Theorem 1. *Given a linear transformation of the plane, $\mathbf{L}$, there exists a unique (up to a reflection) similarity transform of space, $\mathbf{U}$, such that $\mathbf{L}\mathbf{v} \cong \mathbf{U}\mathbf{v}^*$ for any two-dimensional vector $\mathbf{v}$, where $\mathbf{v}^* = (x, y, 0)$ for any $\mathbf{v} = (x, y)$, and $\mathbf{v} \cong \mathbf{w}$ iff $\mathbf{v} = (x, y)$ and $\mathbf{w} = (x, y, z)$.*

The structure of the equivalence between $\mathbf{L}$ and $\mathbf{U}$ stated in the theorem is,

$$\begin{tikzcd} \mathbf{V}^2 \arrow[r, "\mathbf{L}"] \arrow[d, "*"] & \mathbf{V}^2 \arrow[d, leftrightarrow, "\cong"] \\ \mathbf{V}^3 \arrow[r, "\mathbf{U}"] & \mathbf{V}^3 \end{tikzcd}$$

where $\mathbf{V}^2$ and $\mathbf{V}^3$ are two- and three-dimensional vector spaces, respectively. The geometrical interpretation of $\mathbf{U}$ is as a rotation and scale of two basis vectors (defining a plane) so that their image under orthographic projection is the same as applying $\mathbf{L}$ to the basis vectors, as shown in Figure 3.

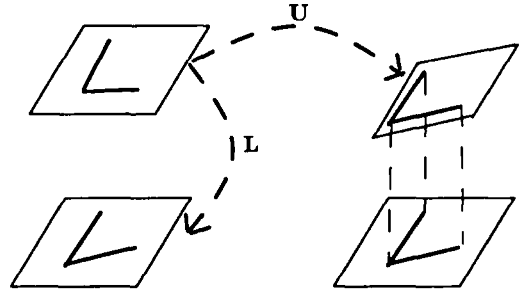


Figure 3. The geometrical interpretation of $\mathbf{U}$.

Proof. Clearly some three-dimensional transformation $\mathbf{U}$ must exist that satisfies the definition of the theorem (for instance just embed $\mathbf{L}$ in the upper-left part of a 3×3 matrix, with the remaining entries all zero). What must be shown is that there is always a $\mathbf{U}$ that is a similarity transformation, and that this $\mathbf{U}$ is unique up to a reflection. In order for $\mathbf{U}$ to be a similarity transform, it must satisfy the two properties of Definition 1. We show that these properties are equivalent to two equations in two unknowns, and that these equations always have two solutions differing in sign. Thus $\mathbf{U}$ always exists and is unique up to a reflection corresponding to the sign ambiguity.

Let $\mathbf{e}_1$ and $\mathbf{e}_2$ be orthonormal vectors in the plane, with

$$\mathbf{e}'_1 = \mathbf{L}\mathbf{e}_1$$
$$\mathbf{e}'_2 = \mathbf{L}\mathbf{e}_2.$$

If $\mathbf{v}_1 = \mathbf{U}\mathbf{e}^*_1$ and $\mathbf{v}_2 = \mathbf{U}\mathbf{e}^*_2$, then by the definition of $\mathbf{U}$ we have,

$$\mathbf{v}_1 = \mathbf{e}'_1 + c_1\mathbf{z},$$
$$\mathbf{v}_2 = \mathbf{e}'_2 + c_2\mathbf{z},$$

where $\mathbf{z} = (0,0,1)$, and c_1 and c_2 are constants.

$\mathbf{U}$ is a similarity transformation iff

$$\mathbf{v}_1 \cdot \mathbf{v}_2 = 0$$
$$\|\mathbf{v}_1\| = \|\mathbf{v}_2\|,$$

because $\mathbf{e}^*_1$ and $\mathbf{e}^*_2$ are orthogonal and of the same length.

From

$$\mathbf{v}_1 \cdot \mathbf{v}_2 = 0,$$
$$(\mathbf{e}'_1 + c_1\mathbf{z}) \cdot (\mathbf{e}'_2 + c_2\mathbf{z}) = 0,$$
$$\mathbf{e}'_1 \cdot \mathbf{e}'_2 + c_1 c_2 = 0,$$

and hence

$$c_1 c_2 = -\mathbf{e}'_1 \cdot \mathbf{e}'_2.$$

The right side of this equality is an observable quantity, because we know $\mathbf{L}$ and can apply it to $\mathbf{e}_1$ and $\mathbf{e}_2$ to obtain $\mathbf{e}'_1$ and $\mathbf{e}'_2$. Call $-\mathbf{e}'_1 \cdot \mathbf{e}'_2$ the constant k_1.

In order for

$$\|\mathbf{v}_1\| = \|\mathbf{v}_2\|,$$

it must also be that

$$\|\mathbf{v}_1\|^2 = \|\mathbf{v}_2\|^2,$$
$$\|\mathbf{e}'_1\|^2 + {c_1}^2 = \|\mathbf{e}'_2\|^2 + {c_2}^2,$$
$${c_1}^2 - {c_2}^2 = \|\mathbf{e}'_2\|^2 - \|\mathbf{e}'_1\|^2.$$

Again the right side of the equality is observable, call it k_2.

It remains to be shown that these two equations,

$$c_1 c_2 = k_1$$
$${c_1}^2 - {c_2}^2 = k_2,$$

always have a solution that is unique up to a sign ambiguity. Substituting

$$c_2 = \frac{k_1}{c_1}$$

into the latter equation and rearranging terms we obtain

$${c_1}^4 - k_2 {c_1}^2 - {k_1}^2 = 0,$$

a quadratic in ${c_1}^2$. Substituting x for ${c_1}^2$ and using the quadratic formula yields

$$x = \frac{1}{2}(k_2 \pm \sqrt{{k_2}^2 + 4{k_1}^2}).$$

A quadratic always has one or two solutions. We are only interested in positive solutions for x, because $c_1 = \sqrt{x}$. There can only be one positive solution, as $4{k_1}^2 \geq 0$, and thus the quantity inside the square root is $\geq k_2$. Hence there are exactly two real solutions for $c_1 = \pm\sqrt{x}$. For the positive and negative solutions to c_1 there are corresponding solutions of opposite sign to c_2, because $c_1 c_2 = k_1$.

The equation for x does not have a solution when $c_1 = 0$, because the substitution for c_2 is undefined. When $c_1 = 0$, however, $c_2 = \pm\sqrt{-k_2}$, which always has two solutions because $k_2 \leq 0$. Recall

$$\|\mathbf{e}'_1\|^2 + {c_1}^2 = \|\mathbf{e}'_2\|^2 + {c_2}^2,$$

therefore, because $c_1 = 0$,

$$\|e'_1\|^2 \geq \|e'_2\|^2,$$

and hence

$$k_2 = \|e'_2\|^2 - \|e'_1\|^2 \leq 0.$$

Thus there are always exactly two solutions for c_1 and c_2, differing in sign. These equations have a solution iff the similarity transform, $\mathbf{U}$, exists. So there are always exactly two solutions for $\mathbf{U}$ which differ in the sign of the $\mathbf{z}$ component of $\mathbf{x}'$, where $\mathbf{x}' = \mathbf{U}\mathbf{x}$, and hence the sign difference corresponds to a reflective ambiguity in $\mathbf{U}$. ∎

We can now prove Theorem 2, the major result of this section.

Theorem 2. *Given three non-colinear points $\mathbf{a}_m$, $\mathbf{b}_m$, and $\mathbf{c}_m$ in the plane, and three corresponding points $\mathbf{a}_i$, $\mathbf{b}_i$, and $\mathbf{c}_i$ in the plane, there exists a unique similarity transformation (up to a reflection), Q, such that $Q(\mathbf{a}^*_m) \cong \mathbf{a}_i$, $Q(\mathbf{b}^*_m) \cong \mathbf{b}_i$, and $Q(\mathbf{c}^*_m) \cong \mathbf{c}_i$, where $\mathbf{v}^* = (x,y,0)$ for any $\mathbf{v} = (x,y)$, and $\mathbf{v} \cong \mathbf{w}$ iff $\mathbf{v} = (x,y)$ and $\mathbf{w} = (x,y,z)$. The transformation Q is a three-dimensional translation, rotation, and a scale factor.*

Proof. From Lemma 1 there is a unique affine transformation such that $A(\mathbf{a}_m) = \mathbf{a}_i$, $A(\mathbf{b}_m) = \mathbf{b}_i$, and $A(\mathbf{c}_m) = \mathbf{c}_i$. By the definition of an affine transformation, A can consists of two components, a translation vector $\mathbf{b}$ and a linear transformation $\mathbf{L}$. Because A is uniquely defined by three pairs of points, we can choose one of the three points pairs to define $\mathbf{b} = \mathbf{a}_m - \mathbf{a}_i$. Then $\mathbf{L}$ is the linear transformation such that $\mathbf{L}\mathbf{b}_m - \mathbf{b} = \mathbf{b}_i$ and $\mathbf{L}\mathbf{c}_m - \mathbf{b} = \mathbf{c}_i$. Using $\mathbf{b}_m - \mathbf{b}_i$ or $\mathbf{c}_m - \mathbf{c}_i$ to define $\mathbf{b}$ is equivalent, because the resulting vector and linear transformation must specify the same affine transformation, A.

Given $\mathbf{L}$, by Theorem 1 there is a unique (up to a reflection) rotation and scale, $\mathbf{U}$, such that $\mathbf{U}\mathbf{v} \cong \mathbf{L}\mathbf{v}$ for all two-dimensional vectors $\mathbf{v}$. Combining $\mathbf{b}$ and $\mathbf{U}$, specifies a unique similarity transformation Q consisting of translation, rotation, and scale such that $Q(\mathbf{a}_m) \cong \mathbf{a}_i$, $Q(\mathbf{b}_m) \cong \mathbf{b}_i$, and $Q(\mathbf{c}_m) \cong \mathbf{c}_i$. ∎

Note that it is not always necessary to explicitly compute the three-dimensional transformation Q in order to transform a model to an image. In the case that the model is planar, the affine transformation A is sufficient to map points from the model plane to the image plane.

2.3 Computing the Transformation

The previous section established the existence and uniqueness of the transformation Q mapping model to image points (and the corresponding affine transformation A). This section shows how to compute Q (and A) given three pairs of points (a_m, a_i), (b_m, b_i) and (c_m, c_i), where the image points are in two-dimensional sensor coordinates and the model points are in three-dimensional object coordinates.

Step 0. Rotate and translate the model so that a'_m is at the origin $(0,0,0)$, and b'_m and c'_m are in the $z = 0$ plane. Note that this operation can be performed offline for each triple of model points.

Step 1. Translate the coordinates of the image points b_i and c_i in terms of the new origin a_i, calling the resulting points b'_i and c'_i.

Step 2. Solve for the linear transformation

$$\mathbf{L} = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

given by the two pairs of equations in two unknowns

$$ax_b + by_b = x'_b$$
$$ax_c + by_c = x'_c,$$

and

$$cx_b + dy_b = y'_b$$
$$cx_c + dy_c = y'_c,$$

where $(x_b, y_b) = b'_m$, $(x_c, y_c) = c'_m$, $(x'_b, y'_b) = b'_i$, and $(x'_c, y'_c) = c'_i$. These first two steps yield the affine transformation, A.

Step 3. Apply $\mathbf{L}$ to the orthogonal axes $(1,0)$ and $(0,1)$ to obtain $\mathbf{e}'_1 = (a, c)$ and $\mathbf{e}'_2 = (b, d)$.

Step 4. Solve for c_1 and c_2, the z components of $\mathbf{v}_1$ and $\mathbf{v}_2$, using

$$c_1 = \pm\sqrt{\frac{1}{2}(w + \sqrt{w^2 + 4(\mathbf{e}'_1 \cdot \mathbf{e}'_2)^2})},$$

where $w = \|\mathbf{e}'_2\|^2 - \|\mathbf{e}'_1\|^2$, and

$$c_2 = \frac{-\mathbf{e}'_1 \cdot \mathbf{e}'_2}{c_1}.$$

Step 5. Define

$$\mathbf{v}_1 = \mathbf{e}'_1 + c_1\mathbf{z},$$
$$\mathbf{v}_2 = \mathbf{e}'_2 + c_2\mathbf{z},$$

and

$$\mathbf{v}_3 = \mathbf{v}_1 \times \mathbf{v}_2.$$

The rotation matrix, R, is defined by the three equations

$$\mathbf{u}_1 = \mathbf{R}(1,0,0)^{\mathrm{T}},$$
$$\mathbf{u}_2 = \mathbf{R}(0,1,0)^{\mathrm{T}},$$
$$\mathbf{u}_3 = \mathbf{R}(0,0,1)^{\mathrm{T}},$$

where $\mathbf{u}_1$ is the unit vector in the direction $\mathbf{v}_1$, and similarly for $\mathbf{u}_2$ and $\mathbf{u}_3$.

If rotations in terms of Euler angles are needed, then the angles α, β, and γ can also be defined,

$$\alpha = \arctan\frac{y_3}{x_3},$$
$$\beta = \arccos\frac{z_3}{\|\mathbf{v}_3\|},$$

where $\mathbf{v}_3 = (x_3, y_3, z_3)$, and

$$\gamma = \angle\mathbf{v}_2\mathbf{q},$$

where $\mathbf{q}$ is the intersection of the $\mathbf{v}_1 \times \mathbf{v}_2$ plane with the $x - y$ plane, which can be computed by rotating the y-axis. $(0,1)$ about the origin by α. These Euler angles define the rotation matrix,

$$R = \begin{bmatrix} \begin{array}{c}\cos\alpha\cos\beta\cos\gamma \\ -\sin\alpha\sin\gamma\end{array} & \begin{array}{c}-(\cos\alpha\cos\beta\sin\gamma \\ +\sin\alpha\cos\gamma)\end{array} & \cos\alpha\sin\beta \\ \begin{array}{c}\sin\alpha\cos\beta\cos\gamma \\ +\cos\alpha\sin\gamma\end{array} & \begin{array}{c}-\sin\alpha\cos\beta\sin\gamma \\ +\cos\alpha\cos\gamma\end{array} & \sin\alpha\sin\beta \\ -\sin\beta\cos\gamma & \sin\beta\sin\gamma & \cos\beta \end{bmatrix}.$$

Step 6. Finally, the scale factor $S = \|\mathbf{v}_1\|$.

Note that the other solution, where c_1 and c_2 are both negated, corresponds to reflecting the model about the plane defined by the three model points.

This method of computing the alignment transformation is relatively fast, because it involves a small number of terms, none of which are more than quadratic. My implementation on a Symbolics 3650 takes about 8 milliseconds to compute the transformation.

3 Recognition Using Alignment

We have implemented a 3D from 2D recognition system called ORA (for Object Recognition by Alignment, pronounced "aura"). The ORA system uses one or two pairs of model and image features to compute possible alignments of solid objects with a two dimensional image. The system first extracts features from the intensity edges in an image. These features are then matched against model features, and used to solve for possible alignments of a model with the image. Each alignment is scored by projecting the aligned model features into the image, and counting the number of projected features for which there is a nearby image feature of the same type and similar orientation. Alignments that account for a high percentage of a model's features are kept as correct matches.

ORA uses three types of primitive features: straight edges, corners, and arcs. Groups of connected or nearly-connected primitive features are combined to form alignment features. Each alignment feature defines either a triple of points or an oriented point. Corresponding alignment features in a model and an image are used alone or in pairs to recover the three-dimensional transformation from a model to an image.

A grey-level image is first processed using a Canny operator [3] to extract intensity edges. Edge pixels are chained into edge contours by following unambiguous 8-way neighbors. At any ambiguity point, new chains are started. Each chain is segmented at inflection points and at the ends of low-curvature regions, as described in [8]. Segments are categorized into the primitive features: straight edge, corner, or arc. A segment that can be well approximated by a single straight line is classified as a straight edge. A segment that has a local high curvature portion is classified as a corner. The remaining segments are classified as arcs.

The Class I features, those defining three points, are shown in Figure 4. The features are: i) an edge with two partial edges, ii) a corner with two complete edges, and iii) an arc with two partial arcs (or edges). The Class II features, those defining an oriented point, are shown in Figure 5. The features are: i) a corner with two partial edges, and ii) an inflection joining two arcs.

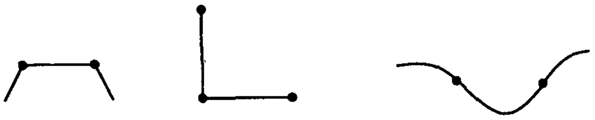


Figure 4. Class I features define three points. A single matching feature is sufficient for alignment.

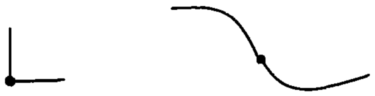


Figure 5. Class II features define an oriented point. A pair of matching features are sufficient for alignment.

A model consists of the three-dimensional locations of the edges of its surfaces. A contour is represented by a chain of three-dimensional edge-pixel locations. Properties of a surface itself are not used (e.g., the curvature of the surface), only the bounding contours. Only the set of surfaces visible from a given viewpoint are used to form a model. Thus for some objects, several models from different views are needed. For objects whose shape is defined mainly by surface characteristics, rather than by the edges of surfaces, this modeling technique is insufficient. For most manmade objects, however, the models appear to be adequate.

3.1 The Matching Algorithm

After the alignment features are extracted from an image, all pairs of Class I model and image features are used to solve for potential alignments, and each alignment is verified using all the model and image features. If an alignment maps sufficiently many model features to image features, then it is accepted as a correct match of an object model to an instance in the image. Each image feature that is accounted for by an accepted alignment is removed from further consideration by the matcher. The exact algorithm is given below, where the variable ALIGNMENTS is used to accumulate the accepted alignments, IMAT is a boolean table indicating the image features that have been matched to some model feature, and MCLASS1 and ICLASS1 are the Class I model and image features. The procedure ALIGN1 computes the possible alignments from a pair of Class I model and image features, VERIFY finds all matching model and image feature pairs given an alignment, GOOD-ENOUGH checks that there are sufficiently many feature pairs, and PAIR-IFEAT is a selector that returns the image feature of a feature pair.

```
for MFEAT in MCLASS1
  do for IFEAT in ICLASS1
    do if not IMAT(IFEAT)
      then for ALIGNMENT in ALIGN1(MFEAT,IFEAT)
        do MATCH ← VERIFY(ALIGNMENT)
          if GOOD-ENOUGH(MATCH)
            then put ALIGNMENT on ALIGNMENTS
              for PAIR in MATCH
                do IFEAT ← PAIR-IFEAT(PAIR)
                  IMAT(IFEAT) ← true
                od
        od
      od
  od
```

Once the Class I model and image features have been exhausted, all pairs of Class II model and image features that have not already been accounted for (are not marked in IMAT) are used to solve for potential alignments.

The alignment computation specifies two possible transformations, that differ by a reflection of the model about the three model points. A further ambiguity may be introduced by not knowing the exact correspondence between the points in the model feature and the points in the image feature. For instance, for a feature that defines two points and two orientations, there are two possible correspondences between the a feature and the image feature. Sometimes it is possible to discard one of these two possible correspondences based on the kind of partial edges (arc versus straight) at each end. Thus for a pair of Class I model and image features (procedure ALIGN1) there are either two or four possible transformations from the model to the image, whereas for two pairs of Class II model and image features (procedure ALIGN2) there are two possible alignments.

To verify an alignment (procedure VERIFY), each model feature (both Class I and Class II) is transformed into image coordinates to determine if there is a corresponding image feature. The image features are stored in a table according to position and orientation, so only a small number of image features are considered for each model feature. The verification procedure requires a good correspondence between a model feature and an image feature, not just a high correlation of the model with the image. For example, an image edge contour that continues beyond the end of a model edge contour is not a good match, whereas one that coterminates is. Similarly, if an image edge crosses a model edge, then that model edge is not well matched. This is important in complex images, of the kind the ORA system has been tested on [9].

In the current implementation, a model feature has a match if there is a corresponding image feature of the same type, approximate position (within 10 pixels), and approximate orientation (within $\pi/10$ radians). A given alignment is verified if more than a certain percentage (currently half) of the transformed model features have a corresponding image feature.

When an alignment matches a model to an image, each image feature that is matched to a model feature is taken to be accounted for by that moiel, and is removed from further consideration by the matcher (by marking it in IMAT). This can greatly reduce the set of matches considered, but at the cost of possibly missing a match in the rather unlikely event that the features from a given instance of an object in an image are incorrectly incorporated into verified alignments of other objects.

The worst case running time of the matching process is $O(m^3 i^2)$ for m model features and i image features. This is because all the features may be of Class II, and it may be necessary to consider matching each pair of model features against each pair of image features, which is $m^2 i^2$. Scoring each alignment then involves transforming each model feature to image coordinates, which is another m operations. In practice, however, it is not necessary to consider all pairs of features, because a given alignment will eliminate other image features from consideration.

Each alignment computation is independent of all the others, so the alignments can all be computed in parallel on a massively parallel machine such as the connection machine [7]. This computation is constant time as long as the number of alignments to be computed does not exceed the number of processors. All the alignments can then be verified in parallel, which will take $O(m)$ time, assuming there is one processor per alignment computation, because each of the m model features must be considered for each alignment.

3.2 Examples

The recognizer has been tested using some images of simple polyhedral objects in relatively complex scenes, taken under normal lighting conditions in offices or laboratories. While the representation is not constrained to polyhedra, it is currently difficult to enter the three-dimensional coordinates of non-polyhedral objects in order to form models.

Figure 6 and Figure 7 show some examples of the recognizer. Part i) of each figure shows the grey-level image, part ii) shows the Canny edges, part iii) shows the primitive features (straight edges are in bold and corners are marked by dots), and part iv) shows all the recognized instances of the models in the image.

There are several hundred alignment features in each of these images, so many thousand possible positions and orientations of each object model were considered. All the hypotheses that survived verification are shown in part iv) of the examples. The matching time (after feature extraction) is 2-3 minutes per image on a Symbolics 3650. About one third of the time is spent computing alignments, and the remainder is taken up transforming model features and looking up corresponding image features for verification.

As can be seen in the examples, the determination of position and orientation is quite good, but does not bring the entire model into exact correspondence with the image. For tasks where a more exact alignment is required, a least squares or other error minimization procedure may be used to improve the estimate of position and orientation. This refined pose estimate can be computed using perspective projection, rather than weak perspective, for even higher accuracy. The idea is to use alignment under weak perspective to find nearly correct positions, because it is relatively computationally efficient, and then to perform less efficient but more accurate positioning only for verified alignments.

4 Summary

We have shown that under the weak perspective imaging model, three corresponding model and image points are necessary and sufficient to align a rigid solid object with an image (up to a reflective ambiguity). We give a closed form solution for computing the alignment from any triple of model and image points. In contrast, other 3D from 2D recognition systems generally use approximate solution methods rather than solving directly for all possible transformations. Furthermore, systems that use a perspective projection model of imaging require at least six pairs of corresponding points to compute the transformation.

The ORA system uses one or two corresponding model and image features to align a model with an image, and then verifies the alignment by transforming the model features into image coordinates. The features are based on a small number of local edge properties, making them relatively reliable under sensor error and partial occlusion. The fact that we have identified the minimum amount of information necessary to recover position and orientation makes it possible to specify features that are both recognizable in real images and sufficient to align a model with an image.

Two classes of features are used, those that define three points, and those that define an oriented point. A single three-point feature, or a pair of oriented-point features, is sufficient to hypothesize a potential position and orientation of a model with respect to an image. Thus there are at most $O(m^2i^2)$ possible alignments to consider for m model features and i image features. Verifying each alignment then takes $O(m)$ time to transform each model feature to image coordinates. We have shown some examples of recognizing objects in relatively complex indoor scenes, under normal lighting conditions.

References

1. Besl, P.J. and Jain, R.C. (1985) Three-Dimensional Object Recognition, *ACM Computing Surveys* Vol. 17, No. 1 pp. 75-154.

2. Brooks, R.A. (1981) Symbolic Reasoning Around 3-D Models and 2-D Images, *Artificial Intell.*, Vol. 17, pp. 285-348.

3. Canny, J. (1986) A Computational Approach to Edge Detection, *IEEE Trans. Pat. Anal. and Mach. Intel.*, Vol. 8, No. 6, pp. 679-698.

4. Cyganski, D. and Orr, J.A. (1985) Applications of Tensor Theory to Object Recognition and Orientation Determination, *IEEE Trans. on Pat. Anal. and Mach. Intel.*, Vol. PAMI-7, No. 6, pp. 662-673.

5. Efimov, N.V. (1980) *Higher Geometry*, English translation from the Russian, Mir Publishers, Moscow.

6. Fischler, M.A. and Bolles, R.C. (1981) Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography, *Commun. Assoc. Comput. Mach.*, Vol. 24, No. 6, pp 381-395.

7. Hillis, W.D. (1986) *The Connection Machine*, MIT Press, Cambridge, Mass.

8. Huttenlocher, D.P. and Ullman, S. (1987) Object Recognition Using Alignment, *Proceedings of the First International Conference on Computer Vision*, IEEE Computer Society Press, pp. 102-111.

9. Huttenlocher, D.P. (1988) Three-Dimensional Recognition of Solid Objects from a Two-Dimensional Image, Doctoral Dissertation, Department of Electrical Engineering and Computer Science, MIT.

10. Kanade, T. and Kender, J.R. (1983) Mapping Image Properties into Shape Contraints: Skewed Symmetry, Affine Transformable Patterns, and the Shape-from-Texture Paradigm, in J. Beck, et. al. (Eds) *Human and Machine Vision*, Academic Press, Orlando, Fla.

11. Lamdan, Y., Schwartz, J.T. and Wolfson, H.J. (1987) On Recognition of 3-D Objects from 2-D Images, New York University, Courant Institute Robotics Report, No. 122.

12. Lowe, D.G. (1987) Three-Dimensional Object Recognition from Single Two-Dimensional Images, *Artificial Intell.*, Vol. 31, pp. 355-395.

13. Silberberg, T.M., Harwood, D.A. and Davis, L.S. (1986) Object Recognition Using Oriented Model Points, Computer Vision, Graphics, and Image Proc., Vol. 35, pp. 47-71.

14. Thompson, D.W. and Mundy, J.L. (1987) Three-Dimensional Model Matching from an Unconstrained Viewpoint, *Proceedings of the International Conference on Robotics and Automation*, IEEE Computer Society Press, pp. 208-220.

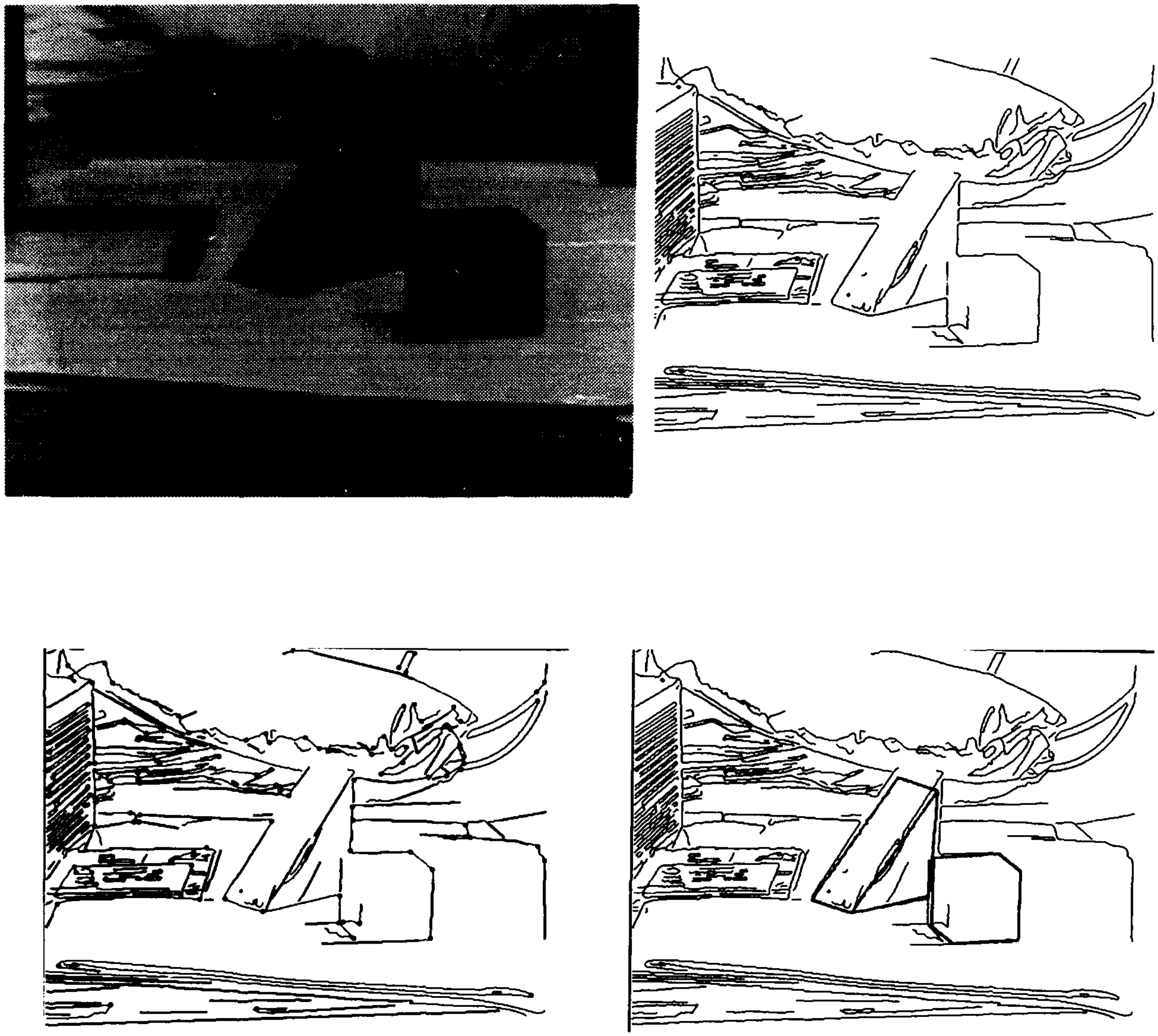


Figure 6. The output of the recognizer: i) grey-level image input, ii) Canny edges, iii) primitive features, iv) recovered instances.

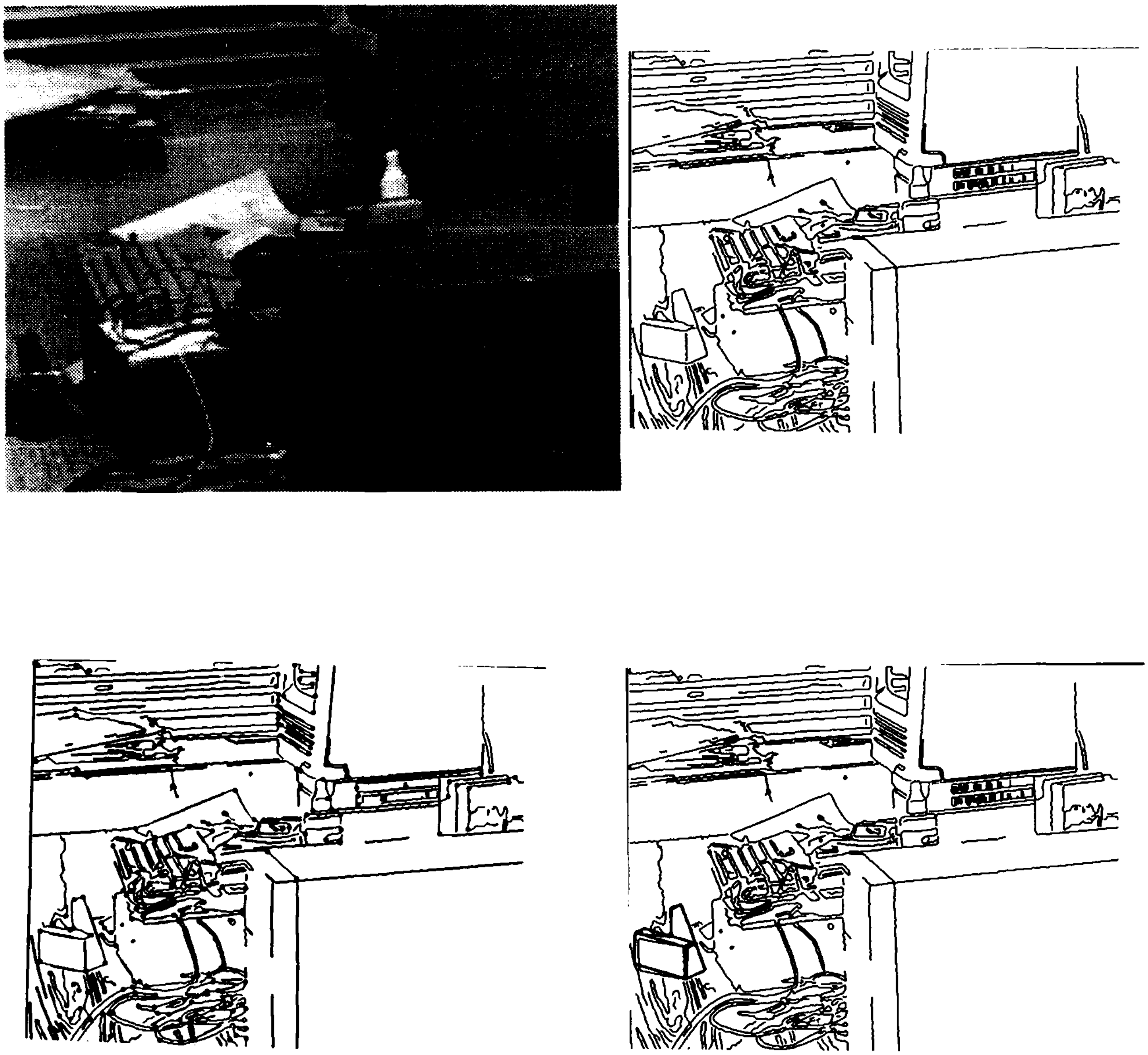


Figure 7. The output of the recognizer: i) grey-level image input, ii) Canny edges, iii) primitive features, iv) recovered instances.

PROJECTIVE INVARIANTS OF SHAPES

Isaac Weiss

Center for Automation Research

University of Maryland

College Park, MD 20742

Abstract

A major goal of computer vision is object recognition, which involves matching of images of an object, obtained from different, unknown points of view. Since there are infinitely many points of view, one is faced with the problem of a search in a multidimensional parameter space. A related problem is the stereo reconstruction of 3-D surfaces from multiple 2-D images. We propose to solve these fundamental problems by using geometrical properties of the visible shape that are invariant to a change in the point of view. To obtain such invariants, we start from classical theories for differential and algebraic invariants not previously used in image understanding. As they stand, these theories are not directly applicable to vision. We suggest extensions and adaptation of these methods to the needs of machine vision. We study general projective transformation, which include both perspective and orthographic projections as special cases.

1. INTRODUCTION

A major goal of any vision system (natural or machine) is to identify objects that are visible in the scene, regardless of their position, orientation or scale relative to the viewer (scale being dependent on distance). The shape of an object is independent of the point of view from which it is seen, namely the viewer-centered coordinates, but its visible image is highly viewer-dependent. This raises the problem of how to separate the intrinsic characteristics of the object from the incidental characteristics of the point of view. When building a data base of objects, for instance, we would like to be able to store only the intrinsic properties of the objects. These intrinsic properties will be invariants, in the sense that they can be calculated from measurements taken in any egocentric coordinate system, but will be independent of the particular system. A simple example is the length of a rod, which is invariant under rotation. In a simple world consisting of rods that lie in a plane, and with images taken orthographically, one can identify a particular rod by measuring its length on the image and comparing it to some data base of rods' lengths. The irrelevant rod's orientation can be ignored.

In current methods of building a data base of shapes [Ballard and Brown, 1984] an image is taken from a particular point of view. Then, to match it to objects seen from a different, unknown view, a search has to be made in the space of the parameters determining the relative point of view, such as rotation angles, scale, and perspective parameters, and a match criterion has to be examined for each point in this parameter space. This is very time consuming. Using invariants can eliminate this search, considerably reducing the amount of time and memory needed to recognize the object.

Other methods attempting to reduce the search size have been employed, for instance using Fourier transforms of closed curves or surfaces, or Hough transforms that match many objects simultaneously. However, besides being restricted in their domain of applicability, none of these methods get rid of all the undesirable parameters; one still has to search in a multidimensional space.

Starting on the most fundamental level, the issue arises of what kind of invariants are useful in vision.

A general consideration is that invariants that are more general, i.e. that stay invariant under a wider class of transformation, are fewer than "weaker" invariants. For instance, the length of the rod, a rotational invariant, is not an invariant under the more general projective transformation. If vision were concerned only with Euclidean transformations, it would make little sense to generalize it to projective transformations. However, many vision problems are concerned with perspective transformation, such as in the case of a picture taken of a planar object, when the planes of the object and the film are not parallel. Projective transformations are the smallest group that contain both perspective and Euclidean transformations as subsets, and hence their importance. Apart from the invariants issue, using projective geometry can unite and simplify the treatment of perspective and orthographic projections, which are usually treated separately [e.g. Horn, 1986].

The case mentioned above of imaging a planar object is often applicable to 3-D objects, since many objects contain planar shapes, such as facets, symmetry planes etc., which are generally projected onto the image perspectively. Thus, 2-D projective transformation and their invariants have to be called on for recognition of many 3-D objects. For planar shapes, one needs only one point of view to measure its projective invariants. The 2-D case will be treated in Section 3.

In the full 3-D case, one has to separate two different issues: "object recognition" and "3-D shape reconstruction". In the former, we deal purely with 3-D. We assume that a visible 3-D shape has already been reconstructed from images and any other clues, and we want to match it to a database of known 3-D objects. The stored shape is likely to differ from the observed one by rotation, translation and dilation transformations, so we need invariants to those, and the demand for projective invariants is stronger than we really need in this case. There exist, however such invariants in 3-D and they are treated in Section 7.

In most applications, however, it is hard to isolate the pure 3-D recogniton problem since it is coupled with the problem of reconstruction from 2-D images. In this case, projective geometry is very helpful in taking advantage of (3-D) contours that might be visible on the object. We can show that in general a 3-D curve, such as a contour, can be reconstructed from *two* 2-D images, without any need for matching or registration of individual points on the curve. By the same method we can recognize the curve and distinguish it from any other curve (up to a projective transformation). (We exclude some degenerate cases.) Other methods do rely on point-by-point matching, which is a difficult and largely unsolved problem. 3-D curves are treated in Sections 3,4,5.

Two main kinds of projective invariant theories have traditionally been studied: differential and algebraic invariants. Halphen [1882], Wilczynski [1906,1907,1908], Lane [1932,1941] and Fubini[1927] dealt with *differential* invariants. These are quantities that are based on local properties of a shape, such as derivatives. For vision, these are good starting points, but several problems arise in applying their theories. First, one has to assume that the functions describing the shape (curve, surface) are sufficiently differentiable, and sometimes up to fourth order derivatives are needed. In practice, however, such derivatives cannot be obtained with satisfactory reliability. Second, as the invariants are local, one has to know the local correspondence between points of the images obtained from different viewpoints. Point correspondence in a very difficult and generally unsolved problem in vision. Thus, the classical theory of differential invariants, while providing a good basis for further development, cannot, as it stands, be applied directly. We study several ways to solve the above problems. Among them: (1) The shape can be expressed as a superposition of some basis functions. The usual methods such as Fourier components or splines are only partially useful, and it is preferable to use basis functions which are themselves invariant. A related method is smoothing or blurring by some filter such as a Gaussian, and again we prefer an invariant filter. As the basis functions are analytic, this superposition can be differentiated reliably. (2) The correspondence and matching problems will be solved by going over to an "invariant space". (3) In the integral features approach (e.g. Amari [1974]) one integrates over the local invariants, and obtains global, or integral, invariants, such as moments. The problem is to find suitable quantities to integrate on. Some Euclidean moment invariants have been discussed by [Reddi, 1981]. Those methods are described in Section 5.

The other classical theory is that of *algebraic* invariants. (See, e.g., [Springer, 1964].) It involves shapes consisting of homogeneous n-dimensional polynomials, such as conics, quadrics, etc, or systems of such components. Under projectivity, some properties of such a shape are preserved. However, most shapes of interest are neither polynomials, nor systems of polynomials, so this theory is not directly applicable to vision. To make use of the theory, we employ a new decomposition of shapes into polynomial *segments*, reported elsewhere [Weiss and Rosenfeld, 1986]. This new decomposition, apart from enabling us to apply the theory of algebraic invariants to general shapes, has numerous other advantages, such as being itself projectively invariant (i.e. the positions of the breakq points on the curve do not change). It can also be obtained very rapidly. These properties also make it advantageous to use our new decomposition, instead of splines, in the case of differential invariants discussed above.

Unlike the differential invariants which change from one point to another, the algebraic ones are constants for whole segments of the shape, thus we shall call them "regional" to distinct them from the local (differential) ones. The method can also be integrated to yield global invariants.

Apart from the geometrical invariants discussed so far, which are intrinsic to the shape, one can make use of extrinsic invariants. These are functions defined on points of the shape, and are obtained from measurements having nothing to do with the geometry, thus being independent of the coordinate system. Examples are the reflectance of a surface, or its color. Shading is also independent of the point of view, provided the reflectance is Lambertian and the lighting remains constant with respect to the object. These quantities are usually local (and thus less sensitive to noise). In trying to make them more global, the geometry of the problem enters. We can apply the method developed for the geometric invariants to make the extrinsic invariants more global. For instance, we can define moments of these quantities in an invariant way. In some cases, such as planar texture, this may be the only reasonable invariant treatment, as the geometry of a plane is not very informative.

Of some use is an additional kind of invariant, the well known cross ratio of four points. The invariance of this ratio has led to the use of invariant coordinates, in which the position of a point is measured by the cross ratio that this point makes with three known reference points. An account of the use of this method in vision can be found in [Duda and Hart, 1973]. However, the problem of finding correspondences between reference points in different views again prevents the wide use of this method, although in some cases statistical methods can be used [Chang, Davis, Dunn, Eklundh and Rosenfeld, 1986].

The next section will briefly review the basic projective geometric tools needed later. The following sections will deal with plane and space curves, developing the various invariants described above. The last sections will deal with surfaces.

2. GEOMETRICAL PRELIMINARIES

2.1 Homogeneous Coordinates

Projective transformations, or projectivities, can be defined geometrically in n-dimens-ional space, and are easiest to visualize for 2-D as a projection by means of a pointlike light source of one plane onto another. The corresponding algebraic expressions are non-linear in Cartesian coordinates (homography). Going over to the so-called homogeneous coordinates yields perspectivities, and makes it possible to develop a rather complete theory of projective invariants.

The homogeneous coordinates of a point (x, y, z) are defined as follows: A point

$$\vec{x} = (x_1, x_2, x_3, x_4)$$

in 4-D space corresponds to (x, y, z) in 3-D space if

$$x = \frac{x_1}{x_4}, \qquad y = \frac{x_2}{x_4}, \qquad z = \frac{x_3}{x_4}$$

Of course this definition is not unique, as the x_i can be multiplied by a common factor and still correspond to the same (x, y, z). In fact, a whole 4-D line corresponds to a 3-D point. (A useful invariant should be unaffected by this multiplication, as well as being unchanged by transformations in the original 3-D space.) For a point with $x_4 = 0$, the division is not possible and such a point does not correspond to a point in Euclidean space. Intuitively, it is a point at infinity. The point (0,0,0,0) is excluded from the space. For projective geometry in the plane, x_3 plays the role of x_4 above.

It can easily be shown that a general projective transformation corresponds to a general *linear* transformation in the homogeneous coordinate space, i.e. to every projectivity in n-D space corresponds a linear matrix in $(n+1)$-D space, defined up to a multiplicative factor.

2.2 Invariants and Covariants

Let us define these concepts involving invariance more accurately, for general transformations (not only projectivities). The geometric shape itself is a fixed entity in space, but its algebraic representation necessitates choosing some coordinates and parameters, and it is their transformation which raises the invariance issue. A shape can be represented by a relation

$$\Phi(a_k, x_i) = 0 \qquad (2.1)$$

The a_k may include parameters which are defined on the shape, or other coefficients. Examples are a parameter t along a curve, or the coefficients of a conic. The x_i are the space coordinates of a point on the shape. The function Φ defines the shape by relating the shape parameters a_k to the space coordinates. The theories of invariance begin by finding a function Φ that preserves its *form* under the transformation of either a_k or x_k in which we are interested. For instance, a conic, which is a second order homogeneous polynomial (*in homogeneous coordinates*), preserves this form under projectivity. In the differential theory, Φ is a linear differential equation. Generally, when the x_i are transformed to $\bar{x}_i$, one substitutes in Φ the x_i as functions of the new coordinates $\bar{x}_i$. After suitable rearrangements one obtains a function of the same form Φ. The same applies to a transformation of a_k.

An invariant is a function of the parameters a_k which, under the transformation, changes in a way that depends only on the transformation constants (but not on a_k or x_i.) I.e., the change, if any, is independent of the shape. This is in general, a *relative* invariant. If there is no change at all, it is an *absolute* invariant.

Often, a relative invariant involves only the Jacobian of a transformation. In this case one defines *a relative invariant of weight* w as a function of the parameters, $I(a_k)$, such that

$$\bar{I}(\bar{a}_k) = J^w I(a_k)$$

where J is the Jacobian of the transformation. When $w = 0$, so that $\bar{I} = I$, the invariant is absolute.

For dealing with integral functionals such as moments or features, we shall need *covariants*. A covariant contains the coordinates x_i as well as a_k. We thus define *a covariant of weight* w *and degree* d as a function of the parameters *and* the coordinates $C(a_k, x_i)$ having the properties

$$C(\bar{a}_k, \bar{x}_i) = J^w C(a_k, x_i)$$

$$\bar{C}(a_k, \lambda x_i) = \lambda^d C(a_k, x_i)$$

The first relation is similar to the one for the invariants. The second means that if all the coordinates x_i are multiplied by a common factor $\lambda(a_k)$, C is multiplied by the same factor, to some power d. This latter transformation property is typical of any homogeneous function of degree d. A covariant of degree zero is an invariant. Φ itself is a covariant.

The geometric invariants for 3-D surfaces are derived from the mathematical representation of the shape in 4-D. The extrinsic invariants are given in 3-D, as functions $f(x, y, z)$. They can be extended to 4-D by substituting for x, y, z their homogeneous equivalents, $x_1/x_4 \cdots$. Thus, f will be constant on the line in 4-D corresponding to a point (x, y, z) in 3-D.

Although in general vision theory is interested in visible surfaces, curves are also very important, for example as discontinuities on surfaces, or occluding contours. The invariance properties of contours will also help us in the stereo reconstruction problem. In Section 3–5 we discuss invariants of curves and their role in vision. In Section 7–8 we similarly discuss invariants of surfaces.

3. DIFFERENTIAL INVARIANTS OF CURVES

3.0 A 1-D Differential Invariant

We first mention a well known one dimensional differential invariant, namely the Schwarzian derivative [Springer 1964]. Consider a particle moving along a straight line, with its position at a time t measured by a (non-homogeneous) coordinate $r(t)$. The Schwarzian derivative $S(r)$ is defined as

$$S(r) \equiv \frac{r'''(t)}{r'(t)} - \frac{3}{2}\left(\frac{r''(t)}{r'(t)}\right)^2$$

and it is invariant under projective transformations of the line and under change in the parameter t, as can be checked directly. Furthermore, the differential equation

$$S(r) = g(t)$$

where $g(t)$ is given, determines the relation $r(t)$ up to (1-D) projectivity.

3.1 General Curves – Introduction

For higher dimensions, the theory is a little less immediate. It is summarized below. We work in homogeneous coordinates.

A curve can be represented parametrically as a function

$$x(t) = x_i(t)$$

where $i = 1,2,3,4$ for space curves, and $i = 1,2,3$ for plane curves. The first step is finding a projective invariant relation that a curve satisfies (the $\Phi = 0$ in Section 2.2). Such a relation has been found in the form of a differential equation whose solution is the curve $x_i(t)$. At first one may question the wisdom of replacing a simple explicit representation of the curve by a differential equation. The advantage can be seen intuitively as follows: when one differentiates a function, certain constants are eliminated. These constants are usually not invariant, while some relations between the derivatives are intrinsic to the curve and are invariant. As a simple example, the equation $x_i'' = 0$ represents all straight lines, regardless of their slope and intercept, and only straight lines. The "straightness" property is invariant under projectivity, but the slope and intercept depend on the coordinates.

It can be shown that a general curve in n-D homogeneous space satisfies the linear differential equation

$$x^{(n)} + \binom{n}{1} p_1 x^{(n-1)} + \binom{n}{2} p_2 x^{(n-2)} + \cdots + p_n x = 0 \qquad (3.1)$$

which is projectively invariant. $x^{(n)}$ means the n-th derivative of $x(t)$. (The same equation is satisfied by each component $x_i(t)$ of the vector $x(t)$, where $p_i(t)$ are scalar functions of t.)

The following theorem is fundamental to the theory of invariants:

Theorem. *The most general (non-degenerate) solution of equation (3.1) is a curve in $n - 1$ dimensional space, determined up to a projective transformation.*

Thus, an equation of the form (3.1) can be identified with a class of curves that differ only by a projectivity, and studying invariants amounts to studying the coefficients p_i of this equation.

Given the parametric representation $x_i(t)$, the coefficients $p_i(t)$ can easily be calculated in the following way. Substitute the n functions $x_i(t)$ one at the time in (3.1). The result is n algebraic equations, at each point t, in the n unknowns $p_i(t)$. Solve this linear system of equations. The solution always exists for a proper curve (i.e. one that does not degenerate to a lower order entity such a point), since the determinant of this system is non-zero. To exlude degeneracy, it is sufficient to demand that the curve does not satisfy any first order differential equation.

Although the coefficients $p_i(t)$ are absolute invariants under projectivities, they still depend on some arbitrary choices that we have made in the representation. First, one has the parameter t. This parameter is chosen arbitrarily and will not necessarily be the same for different observers. Thus we need functions of the p_is that are invariant under transformation of the parameter. Second, the coordinates can be multiplied by a common factor λ. If This factor is constant throughout the shape, there is no problem since its change is a trivial (identity) projectivity, but when it depends on the parameters the invariance is lost. However, some quantities made up of the p_is and their derivatives are invariant and covariant with respect to these transformations, and they will be given next.

3.2. Plane Curves

For a plane curve the projective invariant differential equation is

$$x''' + 3p_1x'' + 3p_2x' + p_3x = 0$$

(A vector equation.) One can define the quantities

$$P_2 = p_2 - p_1^2 - p_1'$$

$$P_3 = p_3 - 3p_1p_2 + 2p_1^3 - p_1''$$

These quantities can be called semi-invariants. They remain unchanged under multiplication of the coordinates by a factor $\lambda(t)$, but not under change of the parameter t.

The invariants are

$$\Theta_3 = P_3 - \frac{3}{2}P_2'$$

$$\Theta_8 = 6\Theta_3\Theta_3'' - 7(\Theta_3')^2 - 27P_2\Theta_3^2$$

The subscripts 3 and 8 of Θ indicate the weights of these invariants. (Under change of the parameter t, they transform as $\bar{\Theta}_w = (d\bar{t}/dt)^w\Theta_w$, where $\bar{t}(t)$ is the new parameter along the curve, and w is the negative of the subscript.) Θ_3 is the only linear one. All other invariants can be derived from Θ_3, Θ_8 and their derivatives. We can thus call the a above two invariants a *complete* set of independent invariants.

Furthermore, the following theorem will be of importance:

Theorem. *The invariants Θ_3, Θ_8 completely determine a plane curve up to a projective transformation.*

Examples of other (dependant) invariants that might be useful are

$$\Theta_{12} = 3\Theta_3\Theta_8' - 8\Theta_8\Theta_3'$$

$$\Theta_{16} = \Theta_3\Theta_{12}' - 4\Theta_{12}\Theta_3'$$

$$\Theta_{21} = 2\Theta_8\Theta_{12}' - 3\Theta_{12}\Theta_8'$$

among which there is the nonlinear relation

$$\Theta_{12}^2 + \Theta_3\Theta_{21} - 2\Theta_8\Theta_{16} = 0$$

We now turn to the covariants. We define

$$z_1 = x' + p_1x$$

$$z_2 = x'' + 2p_1x' + p_2x$$

As these z_i are homogeneous polynomials of degree one, the quantities z_i/x are semi-covariants. The covariants are

$$C_2 = z_1^2 - 2xz_2 - P_2x^2$$

$$C_4 = \Theta_3'x + 3\Theta_3z_1$$

The subscripts of C indicate the weights of the covariants. The degree is the degree of x in the C_i, which are homogeneous. This is a complete set. All other covariants can be expressed as functions of these and invariants.

3.3. Space Curves

We can proceed in a similar fashion to deal with space curves. The differential equation is

$$x'''' + 4p_1x''' + 6p_2x'' + 4p_3x' + p_4x = 0$$

The semi-invariants are

$$P_2 = p_2 - p_1' - p_1^2$$
$$P_3 = p_3 - p_1' - 3p_1p_2 + 2p_1^3$$
$$P_4 = p_4 - 4p_1p_3 - 3p_2^2 + 12p_1^2p_2 - 6p_1^4 - p_1'''$$

The invariants are

$$\Theta_3 = P_3 - \frac{3}{2}P_2'$$
$$\Theta_4 = P_4 - 2P_3' + \frac{6}{5}P_2'' - \frac{6}{25}P_2^2$$
$$\Theta_8 = 6\Theta_3\Theta_3'' - 7\Theta_3'^2 - \frac{108}{5}P_2\Theta_3^2$$

All other invariants are functions of the above and their derivatives. Furthermore, these invariants completely characterize the curve, up to a projective transformation.

For the covariants, we first define

$$z_1 = x' + p_1x$$
$$z_2 = x'' + 2p_1x' + p_2x$$
$$z_3 = x''' + 3p_1x'' + 3p_2x' + p_3x$$

with the semi-covariants being z_i/x.

The covariants are

$$C_2 = 10z_1^2 - 15xz_2 - 12P_2x^2$$
$$C_3 = 10z_1^3 - 3C_2z_1 - 9(5z_3 + 6P_2z_1 + P_3x)x^2$$
$$C_4 = 2\Theta_3z_1 + \Theta_3'x$$

which is also a complete system. We turn now to applying the systems of invariants and covariants for vision purposes.

4. CURVE INVARIANTS IN VISION

The above invariants are ideally suited for solving the problems of matching of shapes stored in a data base to an observed shape, and the stereo reconstruction problem. This is because they contain all the information that is intrinsic to the shape itself, and none of the information that is particular to the particular coordinate system, namely to the particular point of view, which stands in the way of the recognition process.

Since high order derivatives are involved, the formulæ above cannot be used directly. In this section we study the method of overcoming the problem by decomposing the functions into simpler building blocks, which can be differentiated analytically. The advantage of the method (as compared to integral methods described later) is in preserving much of the local characteristic of the shape, making it possible to deal with occluded shapes. It may be a little more complex, however.

As the system of invariants depends non-linearly on the curve's coordinates $x_i(t)$, this linear decomposition will not simplify it. The only purpose of the decomposition is reliably calculating the derivatives. One possible decomposition is the Fourier transform. However, it can make the problem global again, defeating the purpose of handling occluded shapes. Besides, the sines and cosines are not projective invariants. Another possible superposition is of polynomial basis functions, e.g. splines, which are piecewise analytic. The question arises here of the degree of the splines. As the differential equation contains a third derivative, and the invariant functions have at least one more, one can show that at least a quintic spline is needed (for a continuous fourth derivative). The usual splines are not projective invariants. However, since they are polynomials, their theory can readily be extended to one of *homogeneous* polynomials in projective coordinates. In this way they can be turned into algebraic invariants (discused in Section 6). Thus, the homogeneous polynomial splines are particularly suited for representing a visual shape invariantly.

Once the derivatives are known, they can be substituted in the differential equation to find the coefficients $p_i(t)$. This step is quite trivial, as we have a low-order system of linear equations.

It can be simplified even further by choosing the last coordinate, i.e. x_3 in the plane or x_4 in 3-D space, as a constant. This will result in p_3 (or p_4) being identically zero.

In the following step, invariants and covariants can be calculated in a straightforward manner from the p_i, using the formulæ given above. All the invariants given there are relative, i.e. they contain the Jacobian of the transformation. We are interested now only in absolute invariants, and they can be obtained as combinations of the relative ones.

Among the possible absolute invariants of plane curves are

$$\frac{\Theta_3^8}{\Theta_8^4}, \quad \frac{\Theta_3^4}{\Theta_{12}}, \quad \frac{\Theta_8^3}{\Theta_{12}^2}, \quad \frac{\Theta_3^2 C_2}{C_4^2}$$

The first and last quantities are in general independent, as they derive from four independent quantities. There are, however, important degenerate cases, to be discussed later. The others are independent too, in general, but they need the use of higher derivatives. For a space curve, enough independent low order invariants exist to satisfy our purposes.

The final, and most important step is using the above invariants in our specific vision problems. (i) In the pure recognition problem, one has a curve in either 2 or 3-D, and the goal is to match it with a curve that was seen from a different point of view, e.g. a curve stored in a data base. The dimensionality of the curve and the space are unchanged. To do this, one can plot one independent invariant of the curve against another. This will be done both for the stored curve and the observed one. This plot is intrinsic to the curve and remains the same regardless of any projective or parametric transformation that the curve may have undergone. Furthermore, since the invariants completely characterize the curve, it is clear that if the plots obtained from the observed and stored curves are the same, they belong to a similar curve (up to a projective transformation). Thus we have found a way of uniquely *identifying* a curve, viewed in a different coordinate system, without any search in a parameter space.

(ii) In the reconstruction problem, one has a space curve projected onto two planar images. One needs three invariants to characterize a space curve, and these can be plotted in a 3-D space. I.e., to each space curve corresponds an "invariant curve" in another 3-D "invariant" space whose coordinates are the three invariants. Now, given one planar projection of the curve, the 3-D invariants can be determined only up to one parameter (corresponding to the unknown depth). Therefore, for each point on a projected curve, there corresponds a whole (curved) line in the 3-D space of invariants. The entire planar projection thus generates a surface in the invariant space, made up of the lines corresponding to the individual points. Given two planar images of a space curve, one can generate two surfaces in the 3-D invariant space. These surfaces intersect in a curve, which is the invariant curve corresponding to the original space curve. (The intersection always exist if the geometric curve exists.) This invariant curve is sufficient to identify the original curve, similarly to the plot in (i). We have thus been able to solve the stereo reconstruction problem, along with the recognition problem, of a general (non- degenerate) space curve. This was accomplished without a costly search in a multidimensional parameter space, and without facing the problem of pointwise matching or registration of curves observed from different views. The matching is obtained automatically as a result of the intersection.

5. INTEGRAL INVARIANTS OF CURVES

In many cases it useful to characterize a shape by global, integral quantities such as moments, or other "linear features". If one can define such quantities in an invariant way, they can help in identifying objects whose integral quantities are known. An integral quantity of a curve can be defined generally in ordinary coordinates as

$$M = \int I(x,y,z)m(x,y,z)dl(x,y,z)$$

I is some quantity measured on the surface. We will take I to be an invariant, intrinsic or extrinsic. m is a predefined "measuring function", such as a power of x_i if M is to be a moment, and $l(x,y,z)$ is an integration parameter along the curve, such as the arclength. Similar methods apply to convolutions, in which case the integrand will contain a filter or a kernel $G(x,y,z)$.

The above quantities are not invariant, because neither the measuring function m nor the length element dl are. For M to be invariant, the integrand $(Imdl)$ has to be so. Also, the quantities m and l have to be *functions* of the coordinates; otherwise the integration is meaningless. (m can be constant as a special case, but not dl.) Thus, we have to use the covariants in this case, rather than the invariants, which, by definition, do not depend on the coordinates. A convenient general way to obtain covariant functions is to define them as functions of covariant arguments, instead of the coordinates x_i appearing m, l. Consider the covariant vector C_2, defined in Sec 3.3. This quantity is linear in x_i, so functions such as moments defined with powers of C_2 are closely related to the corresponding functions of x_i. Therefore we shall use C_2 to define kind of "covariant coordinates". A minor problem arises from the fact that unlike x, y, z, the covariant is defined in homogeneous coordinates, and one has to make sure that multiplication of these coordinates by a common factor λ will not affect the result (i.e. we need a covariant of degree zero). Our C_2, being homogeneous in x_i, is of degree 1, i.e. multiplication of x_i by λ will result in the multiplication of $C_2(x_i)$ by λ. We can handle this problem by dividing the first three members of the covariant $C_2(x_i)$, namely $C_2(x_{1,2,3})$, by the fourth, $C_2(x_4)$. Doing that, we can define arbitrary covariant functions m and l as functions of the "covariant coordinates":

$$\bar{x} = \frac{C_2(x_1)}{C_2(x_4)}, \qquad \bar{y} = \frac{C_2(x_2)}{C_2(x_4)}, \qquad \bar{z} = \frac{C_2(x_3)}{C_2(x_4)}$$

These quantities are of weight and degree zero.

The function I in the integrand can be chosen as one of the remaining invariants or covariants, or as some extrinsic invariant $f(x,y,z)$, or even as a constant, leaving the covariant measuring functions $m(\bar{x},\bar{y},\bar{z})$ to characterize the curve.

The integral "features" M can now be written as

$$M = \int I(x_i)m(\bar{x},\bar{y},\bar{z})dl(\bar{x},\bar{y},\bar{z})$$

where all arguments are implicit functions of the parameter t.

An alternative way to deal with the non-invariance of the (unmodified) line element $dl(x,y,z)$ is by considering ***densities***, $\rho(x_i)$, of invariant quantities, instead of the quantities $I(x_i)$ themselves. The product $\rho(x,y,z)dl(x,y,z)$ is invariant by definition of the density (namely the average I over a line element dl divided by dl.) A density can be obtained either by differentiating an I (intrinsic or extrinsic), or by measuring the density of some extrinsic quantity that is known to be invariant, such as the number of black dots on a given white element of the shape. This number is invariant, even though the shape element, and thus the dots' density, are not by themselves invariant.

In this approach, the integral features M can be written as

$$M = \int \frac{dI}{dl} m(\bar{x},\bar{y},\bar{z})dl$$

or (not equivalently)

$$M = \int \sum_i \frac{dI}{dx_i} m(\bar{x}, \bar{y}, \bar{z}) dx_i$$

In a variation of the density method, the parameter t could be used in the formulæ above instead of l or x. Also, a combination of an invariant and a density (derivative) of another invariant could be used, i.e.

$$M = \int I_a \frac{dI_b}{ds} m ds$$

where s is any of the possible parameters l, x, t.

The above methods assume that the covariant $C_2(x_i)$ is not identically zero, or constant, in all of its four components. In some degenerate cases it does vanish so that the features M vanish, for an arbitrary measuring function m. This vanishing is in itself an invariant characteristic of the shape and can be used to identify it. An example of this case is a straight line. Intuitively, one can say that there are no geometrical covariants of a straight line, other than constants, so we cannot find features, but this fact in itself says enough about this shape.

Extrinsic invariants become particularly important when the geometry of the shape is degenerate, e.g. it is a line or a plane. If an extrinsic invariant f is given, such as a texture pattern, one would like to measure integral quantities of the shape, e.g. moments. The above methods can be easily adapted to these cases. The idea is to use the extrinsic function f in a new space (f, x_i), in which the curve is represented by both its original geometrical coordinates, x_i, and the new "coordinate" f. For a generic function $f(t)$ the curve will no longer be degenerate and we can use the methods described above to derive non-vanishing integral quantities M. Because the shape has degenerated in dimensionality, adding the new dimension f will not complicate the treatment beyond that of a non-degenerate shape. For example, a texture pattern on a planar shape can be represented as a surface of grey levels in 3-D. It is not necessary, however, to use the full 3-D invariant theory because the f coordinate stays invariant and does not transform.

In the texture case the domain of integration for calculating the moments is not obvious. Some integral characteristics, however, can be defined that are independent of the domain, such as the average anisotropy.

The curve can be degenerate in the (f, x_i) space also. This will happen if the extrinsic function f is constant along a shape which is a straight line (or if one has a constant density in some coordinate system). In this case all possible Ms as defined above will vanish, and that will be the invariant characterization of the shape.

6. ALGEBRAIC INVARIANTS

The theory of differential and integral invariants described above is, in principle, complete and should include the "regional invariants" described below as a special case. However, this "special case" has the advantage of being implementable much more simply and rapidly than the general case, while still being applicable to a wide variety of curves and surfaces. It requires neither finding derivatives nor integrals.

The method is based on the segmentation of a curve (or surface) into segments of conic sections (or patches of quadric surfaces). The following discussion is invariant to replacement of (planar) curves by surfaces, conic sections by quadric patches, etc. This can be done, up to a desired tolerance limit, very rapidly by a method described elsewhere [Weiss and Rosenfeld, 1986]. Very briefly, this segmentation method is based on a well known geometrical theorem, according to which all the midpoints of parallel chords cutting a conic section lie on a straight line. The parallel chords can be implemented in real time as camera scan lines, and the straight line formed by their midpoints can be detected by standard methods such as the Hough transform, which work much more efficiently on a straight line than on the original conic. In short, the problem of conic detection is transformed by this method into a problem of straight line detection.

Unlike the usual segmentation method, this one is projectively invariant. The conic sections remain conic sections after a projective transformation. The break-points, i.e. the intersections between the conic sections, also do not move and correspond to the same points on the curve, when the segmentation is done in a transformed coordinate system (within the tolerance limit). Thus, the curve can be segmented invariantly into a string of conic sections.

It remains to find the invariants of these conic sections. Here we can make use of the classical theory of algebraic invariants [Winger, 1962]. This theory is concerned with finding the invariants of algebraic forms in homogeneous coordinates. By algebraic form one means a homogeneous polynomial in several variables. We are now interested in second order (quadratic) forms, which can be written as

$$f = \sum_{i,j} a_{i,j} x^i x^j$$

where $i = 1,2,3$ in the plane and $i = 1,2,3,4$ in 3-D. This can be interpreted as a matrix $a_{i,j}$ multiplied by the vectors x_i, x_i^T. Without loss of generality, this matrix can be assumed to be symmetric. Equating the form f to zero gives an algebraic representation of a conic section (quadric surface). We then have

$$f = \sum_{i,j} a_{i,j} x^i x^j = 0$$

This is a simple case of the general Φ of Sec. 2.2. Under projective transformation of the coordinates x_i, the form of the above equation does not change, but the coefficients $a_{i,j}$ do. The new coefficients can be easily found by expressing the old coordinates in terms of the new ones and substituting in the form f. However, some combinations of the coefficients are preserved, and are thus invariants. We will now list some of the invariants.

A single quadratic form, namely each conic (quadric), has one invariant, the determinant

$$d = |a_{i,j}|$$

This invariant is of weight two, i.e.

$$\bar{d} = dJ^2$$

where J is the Jacobian of the projective transformation. The Jacobian is the same for all the conic sections, so it is easy to eliminate, e.g. by choosing one of the conics and dividing the d of all other conics by the one of that particular conic. Thus, from say k conics, we have so far obtained $k-1$ absolute invariants (i.e. of weight zero).

Next, ones has simultaneous invariants, i.e. one that are made up of the coefficients of two or more forms. Two conics [quadrics] $a_{i,j} x^i x^j = 0$ and $b_{i,j} x^i x^j = 0$ have the two simultaneous invariants of weight two

$$\alpha = A^{i,j} b_{i,j}, \qquad \beta = B^{i,j} a_{i,j}$$

where $A^{i,j}$, $B^{i,j}$ are the cofactors of the matrices $a_{i,j}$, $b_{i,j}$, and a summation over i, j is implied (by the Einstein summation convention), i.e. when i, j are in both sub- and superscripts). These are in addition to the two single conic invariants, namely

$$a = |a_{i,j}|, \qquad b = |b_{i,j}|.$$

One can show that all other invariants of two conics can be expressed as polynomials in the above four.

We can take pairs of our string of conic sections, and find the simultaneous invariants of each pair at a time. We thus obtain $k(k-1)/2$ of these invariants. They cannot be all independent, however, as the above number is greater than the number of coefficients of the k conics, which is linear in k. Thus, a better strategy would be to calculate the simultaneous invariants of pairs of adjacent conics only.

One can go on to find simultaneous invariants of more than two forms, but again we will encounter interdependency. The Clebsch-Gordan theorem states that a set of algebraic forms has a finite and complete basis of independent invariants, and any polynomial function of these basic invariants is also an invariant. It may be of theoretical interest to try and find all the independent invariants of a set of conics [quadrics]. For practical purposes, however, the above methcd of deriving invariants seems sufficient.

Besides their role as invariants in their own right, the algebraic forms are a good choice as a basis functions for the decomposition used in the differential invariant theory of the previous section. In that case, we have several basis function mingling at each point, while in this section we have one basis function for each curve segment. The decomposition of a curve by the usual splines as basis functions is acceptable, even though it is not invariant, because it is only an intermediate step in obtaining the invariants. But more accurate results will probably be achieved if the decomposition is also invariant. Thus "invariant spline" polynomials are of interest. They are different from the ordinary ones by being multidimensional homogeneous polynomials, but the usual spline theory can easily be extended to them.

7. DIFFERENTIAL INVARIANTS OF SURFACES

We turn now to the description of invariants of surfaces in 3-D space. This section will summarize the mathematics, and the next one will discuss vision applications. The algebraic theory, i.e. the theory of surfaces that can be segmented into quadric patches, is essentially similar to the case of plane curves and has already been described in the last section. The differential and integral theories, however, are more involved. It would be desirable, and is theoretically possible, to derive invariants and covariant as functions of any given parametric representation of a surface. However, this has proven to be a very tedious task, so surface invariants have been derived only in specialized systems of coordinates. This does not detract much from the generality of the invariants, because these special coordinate systems are defined in an invariant way and can be derived from whatever coordinate system one is initially given. One does, however, need to invest some effort in deriving these invariant coordinates, and perhaps the vision application will provide the impetus to develop general expressions for the invariants, that do not use any particular coordinate system.

We first need some definitions adapted from classical differential geometry. One has to be careful, however, to make use only of projective invariant terms. Thus one deals with lines, planes, tangents, intersections, but not with curvatures, distances or angles.

- A *tangent line* at a point P_x on a curve C is a line passing through P_x and one neighboring point that approaches P_x along the curve C (to an infinitesimal distance).
- An *osculating plane* of a curve is a "higher order" tangent, in the sense that it has more contact points with the curve. An osculating plane through a point P_x on a curve C is a plane passing through the point P_x and *two* neighboring points, approaching P_x along C. It can be regarded as the plane most closely approximating the curve C in the vicinity of P_x.

The osculating plane of a straight line is undetermined.

- A *tangent plane* to a surface S at a point P_x is a plane passing through P_x and two neighboring, non-collinear points approaching P_x on the surface S.
- An *asymptotic curve* on a surface is one whose osculating plane coincides with the tangent plane to the the surface at each point of the curve. In Euclidean space, this would be the curve having the direction in which the normal curvature to the surface vanishes, but this definition is not projectively invariant. Asymptotes will be used in the sequel as the coordinates relative to which the invariants are calculated. Examples of asymptotes are the generating lines of cylinders or cones.
- A *developable surface* is the locus of the tangent lines to a given curve. These special surfaces will be excluded from the general treatment that follows as they need special treatment.
- A surface is *ruled* if through every point on it passes a straight line which lies entirely on the surface.
- A *net* is made up of two *families* of curve, both covering the whole surface.

We define a surface in 3-D space with the help of the four dimensional projective vector $x = x_i$, $i = 1,2,3,4$, so that

$$x = x(u,v),$$

or equivalently

$$x_i = x_i(u,v),$$

where $x_i(u,v)$ are analytic functions of the two parameters u, v.

One can derive an equation for the asymptotic net of the surface, based on the first and second derivatives of x with respect to u, v, as follows. One first defines the three determinants

$$\begin{cases} L = (x_{uu}, & x, & x_u, & x_v), \\ M = (x_{uv}, & x, & x_u, & x_v), \\ N = (x_{vv}, & x, & x_u, & x_v). \end{cases}$$

The equation of the asymptotes can now be written as

$$L du^2 + 2M du dv + N dv^2 = 0 \tag{7.1}$$

This equation means that for any two infinitesimally close points lying on the asymptotic line, the difference du in their u coordinate can be calculated, given the difference dv in their v coordinate, and vice versa. The equation is quadratic, with the discriminant $NL - M^2$. We exlude the case in which this discriminant vanishes, and thus ensure having two distinct roots of this quadratic equation (not necessarily real). In other words, for some increment dv between two points on an asymptote, we have two possible increments du. Thus, we have two distinct families of asymptotic curves, which can be used as a coordinate net.

The vanishing discriminant has a geometric meaning, as follows:

Theorem. *A surface is developable if and only if* $NL - M^2 = 0$.

Thus, every non-developable surface has two families of asymptotes forming a net.

We want to transform from the u, v coordinates to the asymptotic ones. The transformation equation of the various quantities will be useful when going over to the asymptotic coordinate system, and will also help demonstrate some invariance properties. We first deal with a general coordinate transformation of (u,v) to the coordinates $(\bar{u},\bar{v})$, i.e.

$$\bar{u} = \bar{u}(u,v), \qquad \bar{v} = \bar{v}(u,v) \qquad (J = \bar{u}_u \bar{v}_v - \bar{u}_v \bar{v}_u \neq 0) \tag{7.2}$$

By standard methods it is easy to show that the derivatives transform as

$$
\begin{aligned}
x_u &= x_{\bar u}\bar u_u + x_{\bar v}\bar v_u, \\
x_v &= x_{\bar u}\bar u_v + x_{\bar v}\bar v_u, \\
x_{uu} &= x_{\bar u\bar u}\bar u_u^2 + 2x_{\bar u\bar v}\bar u_u\bar v_u + x_{\bar v\bar v}\bar v_u^2 + x_{\bar u}\bar u_{uu} + x_{\bar v}\bar v_{uu}, \\
x_{uv} &= x_{\bar u\bar u}\bar u_u\bar u_v + x_{\bar u\bar v}(\bar u_u\bar v_v + \bar u_v\bar v_u) + x_{\bar v\bar v}\bar v_u\bar v_v + x_{\bar u}\bar u_{uv} \\
&\quad + x_{\bar v}\bar v_{uv}, \\
x_{vv} &= x_{\bar u\bar u}\bar u_v^2 + 2x_{\bar u\bar v}\bar u_v\bar v_v + x_{\bar v\bar v}\bar v_v^2 + x_{\bar u}\bar u_{vv} + x_{\bar v}\bar v_{vv}.
\end{aligned}
\tag{7.3}
$$

and the quantities M, N, L transform as

$$
\begin{aligned}
\bar L &= Lu_{\bar u}^2 + 2Mu_{\bar u}v_{\bar u} + Nv_{\bar u}^2, \\
\bar M &= Lu_{\bar u}v_{\bar u} + M(u_{\bar u}v_{\bar v} + u_{\bar v}v_{\bar u} + Nv_{\bar u}v_{\bar v}, \\
\bar N &= Lu_{\bar v}^2 + 2Mu_{\bar v}v_{\bar v} + Nv_{\bar v}^2
\end{aligned}
$$

From the last equations it is easy to show that

$$\bar L\bar N - \bar M^2 = \frac{1}{J^2}(LN - M^2).$$

Thus, an equation $LN - M^2 = 0$, if it holds, is invariant, which is not surprising since the property of being developable is invariant. The asymptotes are also invariants because their equation (7.1) is.

We can now choose $\bar u, \bar v$ as the asymptotes. It is of interest to find an explicit equation for these $\bar u, \bar v$, in contrast to eq. (7.1) which gave the asymptotes as increments in the old coordinates du, dv. The fact that the new coordinates coincide with the asymptotes makes $\bar u$ constant along one asymptote and $\bar v$ constant along the other. In terms of u, v this yields

$$
\begin{aligned}
d\bar u &= \bar u_u du + \bar u_v dv = 0 \\
d\bar v &= \bar v_u du + \bar v_v dv = 0
\end{aligned}
$$

This relation means that the ratio of the derivatives $\bar u_u/\bar u_v$ is the negative inverse of the increments du/dv, Substituting this in equation (7.1) for the asymptotes we find a differential equation for $\bar u, \bar v$:

$$N\bar w_u^2 - 2M\bar w_u\bar w_v + L\bar w_v^2 = 0 \tag{7.5}$$

where $\bar w$ can be either $\bar u$ or $\bar v$. Again, a quadratic equation. So, the asymptotic coordinates $\bar u, \bar v$ on the surface can be found either by solving equation (7.5) or by using equation (7.1) for the increments of u, v along the asymptotes, which may be easier in practice.

From now on we shall work with asymptotic coordinates only, so we can drop the bars and denote them by u, v.

We are now in a position to develop the theory of invariants along lines similar to those used for curves. First, one has to find the differential equation satisfied by the surface. The following theorem holds:

Theorem. *Every non-developable surface, parametrized by its net of asymptotes, satisfies a set of two equations of the form*

$$
\begin{aligned}
x_{uu} + 2ax_u + 2bx_v + cx &= 0, \\
x_{vv} + 2a'x_u + 2b'x_v + c'x &= 0.
\end{aligned}
\tag{7.6}
$$

whose coefficients are scalar functions of u, v. A non-degenerate surface will not satisfy any equation of the form

$$Ax_{u,v} + Bx_u + Cx_v + Dx = 0 \tag{7.7}$$

whose coefficients are not all zero.

Conversely: the solution of such a system of equations is a surface that is determined up to a projective transformation, it is non-developable, does not degenerate to a curve, and the parameters u, v are along its asymptotes.

Thus, the above set of equations, with some particular coefficients, identifies one particular class of surfaces, differing only by projectivity. Studying the projective invariant properties of a surface amounts to studying its set of equations as written above.

For completeness, we mention that the coefficients in the above equations can not be completely arbitrary, but are subject to the integrability conditions

$$
\begin{aligned}
&a_v - b'_u = 0, \\
&a'_{uu} + c'_u - 1a'a_u - 2aa'_u - (a_{vv} + 2b'a_v - 2ba'_v - 4a'b_v) = 0, \\
&b_{vv} + c_v - 2bb'_v - 2b'b_v - (b'_{uu} + 2ab'_u - 2a'b_u - 4ba'_u) = 0, \\
&c'_{uu} - 4ca'_u - 2a'c_u + 2ac'_u - (c_{vv} - 4c'b_v - 2bc'_v + 2b'c_v) = 0.
\end{aligned}
$$

As in the case of the curve, the projective invariance of the differential equation is not enough for our purposes. We need invariance to transformation of the parameters along the asymptotes, and invariance to multiplication of the components x_i by a common factor $\lambda(u, v)$.

A change in the parametrization of the asymptotes, without changing the net itself (which is the only change we will dare to make) can be represented as

$$\bar u = \alpha(u), \qquad \bar v = \beta(v) \tag{7.8}$$

where $\alpha(u)$ depends on u only, and $\alpha(v)$ depends on v only. We can then define the "semi-invariants"

$$a', \qquad b,$$

$$f = c - a_u - a^2 - 2bb', \qquad g = c' - b'_v - b'^2 - 2aa'.$$

and "semi-covariants"

$$
\begin{aligned}
z &= x_u + ax, \\
\rho &= x_v + b'x, \\
\sigma &= x_{uv} + b'x_u + ax_v + \frac{1}{2}(a_v + b'_u + 2ab')x
\end{aligned}
$$

With these notations, we can now write the (relative) invariants as

$$
\begin{aligned}
&a', \qquad b, \\
&h = b^2(f + b_v) - \frac{1}{4}bb_{uu} + \frac{5}{16}b_u^2, \\
&k = a'^2(g + a'_u) - \frac{1}{4}a'a'_{vv} + \frac{5}{16}a'^2_v
\end{aligned}
\tag{7.9}
$$

Upon the above transformation of coordinates, these transform as

$$\bar a' = \frac{\alpha_u a'}{\beta_v^2} \qquad \bar b = \frac{\beta_v b}{\alpha_u^2}$$

$$\bar h = \frac{\beta_v^2 h}{a_u^6}, \qquad \bar k = \frac{a_u^2 k}{\beta_v^6},$$

All other invariants can be derived from those above and their derivatives. Absolute invariants are easily obtained from the relative ones.

More invariants can be obtained by the following process. Define the invariants

$$A = a'b^2, \qquad B = a'^2 b, \qquad H = a'h, \qquad K = bk$$

which transform as

$$\bar A = \frac{A}{a_u^3}, \qquad \bar B = \frac{B}{\beta_v^3}, \qquad \bar H = \frac{H}{a_u^5}, \qquad \bar K = \frac{K}{\beta_v^5},$$

and the following operators:

$$U = \alpha'\frac{\partial}{\partial u}, \qquad V = b\frac{\partial}{\partial v} \tag{11}$$

Then the following functions are also invariant

$$U(B), \qquad V(A), \qquad U(K), \qquad V(H)$$

The process can be repeated iteratively.

The basic covariants are

$$x, \qquad 6Az + A_u x, \qquad 6B\rho + B_v x, \qquad \sigma - a\rho - b'z + ab'x. \quad (12)$$

All other covariants can be obtained from the above (and their derivatives) and invariants.

For most surfaces, the four invariants a', b, k, h retain all the information contained in the surface equations (7.6). More accurately, one can state a "fundamental theorem" regarding surface invariants:

Theorem. *The four invariants a', b, h, k determine a non developable surface up to a projective transformation, if a' and b are not zero. If either a' or b is zero, the surface is ruled. If both vanish, the surface is a quadric.*

(A quadric is a "doubly ruled" surface. All the above invariants vanish in this case.) There is no need to worry about these special cases. A theory for ruled surfaces and for developables has been developed along lines similar to the above, with a slightly different form of the differential equations. For a quadric, the algebraic invariants discused earlier form the complete theory.

We conclude this account of the classical theory of differential invariants of surfaces by summarizing the essential steps involved.

(1) Find the determinants L, M, N of the surface's derivatives.

(2) Solve the first-order equation (7.5) (or (7.1)) for the asymptotic coordinates.

(3) Express the surface derivatives in terms of the asymptotic coordinates u, v, substitute in the differential equations *(7.8), and solve the resulting algebraic equations* for the parameters $a, \cdots, c'$.

(4) Derive the invariants and covariants, as functions of the parameters found in (3), using the given formulas.

8. SURFACE INVARIANTS IN VISION

While the projective theory of space curves can be of great value in both the 3-D reconstruction and the recognition problems, the theory for surfaces is valuable primarily when the 3-D surface is already known, either by earlier reconstruction from 2-D images, or if we have range data. Its advantage is freeing us from the need to find visible contours on the surface.

In practical applications, the steps described above for deriving invariants are easier described than implemented. Even after deriving the invariants at each point, we cannot match them point by point, as we do not know the point correspondence. Some of these methods are direct generalizations of those used for curves, but some are special to 3-D.

Step (1) in the summary of the last section, namely obtaining derivatives, is similar to that for curves. *It can be done by* approximating the surface in a way which relies on information about the surface taken from a reasonably wide region around a point, rather than local information. Thus, instead of trying to differentiate the surface depth function, for instance, we will first approximate this function by splines or "invariant" spline functions, and differentiate this approximation.

Step (2), not relevant for curves, is finding the asymptotic lines. To do this, the differential equation (7.5) or (7.1) has to be solved. Since it is a first order PDE, it can solved by standard methods, e.g. by following it characteristics [John, 1982]. These characteristics are none other than the asymptotic lines.

We may be able to use a short cut, by using some metric properties of the asymptotes. For instance, the normal curvature vanishes in the direction of the asymptotes. Also, the asymptotes bisect the angles between the principal curvature directions. Although curvature and angles are not projective invariants, the end result, i.e. the asymptotes, is.

Step (3) should be relatively easy. We need to find the surface derivatives in the new, asymptotic coordinates. But we do not need to to find derivatives again, as we can use equations (7.3) that transform the derivatives in one coordinate system to another. Substituting these derivatives in the general surface equations (7.6), we obtain a simple linear algebraic system of equations for the six unknowns $a, \ldots, c'$. We have two vector equations (since x, x_μ etc. are known 4-D vectors), amounting to eight equations in all.

In step (4) we have to find the invariants using the quantities $a, \ldots, c'$ found in the previous step. This involves differentiation of these quantities. In the curve case, this could be done analytically, because once we found the coefficients in the approximating functions of step (1), all subsequent steps could be performed analytically. In the surface case, however, step (2), finding the asymptotes, will usually stand in that way as it involves numerical computation. There is no general analytic solution to a nonlinear PDE. Thus, we first have to find the quantities $a, \ldots, c'$ at many points on the surface and then find an approximating function that can be differentiated.

The above procedure finds the invariants at any given surface point. We now have to use them for the recognition, or matching problem. For that we need the absolute invariants, that do not depend on either the projectivity nor the parametrization along the asymptotes, nor on a common factor multiplying the homogeneous coordinates. Of the four basic invariants a', b, h, k, two are absolute. We can obtain more absolute invariants by the process indicated above, if necessary, but this will require more differentiation. In analogy with the plotting method in the curve case, we can now map the surface into a k-D invariant space, each surface point being represented by k *absolute invariants. Since the surface is determined by its* invariants up to a projective transformation, the entity obtained in the invariant space will *uniquely* represent the class of surfaces differing only by such projectivity. Any other kind of difference between surfaces will cause a difference in their mapping in the invariant space. A matching algorithm between two projectively different surfaces can thus consist of simply calculating the image of the mapping of two surfaces onto the invariant space, and matching the two images. There is no need for a search in any parameter space. For a general projective transformation of 3-D shape, this could have been a 15-D search space!

The correspondence problem can now be solved as a byproduct. Surface points that map to the same place in the invariant space are likely to correspond to each other. The mapping is not one-to-one, so some ambiguities will arise, but they can be resolved by adding more points. It may be of interest to investigate the minimum conditions for determining a unique point correspondence.

Integral invariants can be obtained in the same fashion as for curves. The covariant integration coordinates $\bar{x}, \bar{y}, \bar{z}$ are now functions of the asymptotic coordinates, and the integral invariant M is defined as (ds being a surface element)

$$M = \int I(x_i) m(\bar{x}, \bar{y}, \bar{z}) ds(\bar{x}, \bar{y}, \bar{z})$$

where all arguments are implicit functions of the parameters u, v.

We have a wider choice in defining our covariant coordinates $\bar{x}, \bar{y}, \bar{z}$. According to eq. (7.12), we have four sets of vector covariants, including the asymptotic coordinates x_i themselves, each of which can be defined as our integration coordinates by dividing the first three components of the vector by the fourth. In this way, these components become invariant to multiplying x_i by a common factor and M becomes an absolute invariant, for every measuring function $m(\bar{x}, \bar{y}, \bar{z})$. By choosing convenient measuring functions, such as powers of $\bar{x}, \bar{y}, \bar{z}$, one can characterize the surface invariantly by a small number of properties (features). *As for curves, one can generalize the treatment of* surfaces by the use of extrinsic invariants, such as shading, color, or texture characteristics. The possibility of using *densities* also extends easily to surfaces.

9. SUMMARY

We have surveyed theories of projective invariants and suggested methods of applying them in vision. The vision problems that can be solved from these theories are very general and fundamental ones, such as 3-D reconstruciton from 2-D images and matching of images observed from different points of view. Unlike other method, we did not have to deal with the difficult problems of search in parameter space and of finding point correspondence. In subsequent papers two lines of development will be explored: (i) implementation; (ii) finding smaller subgroups of the projective transformation (such as Euclidean ones), which have their own invariants, in addition to the general projective ones.

Acknowledgments

The author thanks Rand Waltzman for fruitful discussions.

REFERENCES

Ballard and Brown, *Computer Vision*, Prentice Hall, 1982.

Chang, S., L.S. Davis, S.M. Dunn, J.-O. Eklundh, A. Rosenfeld [1987], Texture Discrimination by Projective Invariants, Pattern Recognition Letters, V, 337.

Clebsch, *Theorie der binären algebraischen Formen*, sechster Abschnitt.

Do Carmo, M.P. [1976], *Differential Geometry of Curves and Surfaces*, Englewood Cliffs, NJ: Prentice-Hall.

Duda, R.O. and Hart, P.E. [1973], *Pattern Recognition and Scene Analysis.* New York: Wiley.

Fubini e Čech [1927], *Geometria proiettiva differenziale*, Bologna: Zanichelli.

Halphen, Mémoires des Savants Étrangers, vol. 28, 2nd series, 1882.

Horn, B.K.P., *Robot Vision,* MIT Press, 1986.

John, F. [1982], *Partial Differential Equations*, New York: Springer Verlag.

Lane, E.P. [1932], *Projective Differential Geometry of Curves and Surfaces*, The University of Chicago Press.

Lane, E.P. [1941], *A Treatise on Projective Differential Geometry*, The University of Chicago Press.

Springer, C. E. [1964], *Geometry and Analysis of Projective Spaces*, San Francisco: Freeman.

Weiss, I. and A. Rosenfeld [1986], Shape Representation by Quadric Surfaces. Unpublished.

Wilczynski, E.J [1906], *Projective Differential Geometry of Curves and Ruled Surfaces*, Leipzig: Teubner.

Wilczynski, E.J [1907], Projective Differential Geometry of Curved Surfaces (First Memoir), Amer. Math. Soc. Trans., VIII, 233.

Wilczynski, E.J [1908], Projective Differential Geometry of Curved Surfaces (Second Memoir), Amer. Math. Soc. Trans., IX, 79.

Winger, R.M. [1962], *An Introduction to Projective Geometry*, New York: Dover.

An optimal algorithm for the derivation of Shape from Shadows

MICHAEL HATZITHEODOROU
JOHN KENDER

Department of Computer Science
Columbia University
New York, NY 10027

Abstract. We study the problem of recovering a surface slice from the shadows it casts on itself when lighted by the sun at various times of the day. The problem is formulated and solved in a Hilbert space setting. The spline algorithm interpolating the data that result from the shadows is constructed. This algorithm is optimal in terms of the approximation error and has low cost. We implement the optimal error algorithm and show a series of test runs. In addition another modified version of the algorithm that improves the cost considerably is shown. This version is suited for parallel computation with further reductions in the cost of the solution.

1. Introduction.

In computer vision the surface reconstruction problem is of crucial importance in the process of recovering the scene characteristics from the 2-dimensional image created by the camera.

In the various *Shape from X* algorithms, different methods of recovering a surface have been proposed. In this paper we will use a new approach for the reconstruction of the surface shape. Namely, we will define and solve the *Shape from Shadows* problem. In this problem the shadows created by a light falling on a surface will be used to recover the surface itself.

Very little work has been done using shadows for the reconstruction of the surface shape. The only work we are aware of is the one proposed in [5] where the problem is solved using a relaxation method.

Shadows are a very strong piece of information. The process that uses shadows is not affected by texture or by surface reflectance. Furthermore, our imaging system does not need a grey scale or color capabilities; it is sufficient for it to be able to distinguish between black and white. Also, noise in the form of bright spots inside a dark area and vice-versa can be filtered out easily. From the above, it is evident that shadows yield a powerful tool to be used in the reconstruction process.

Our problem will be the following. We have a surface which is lighted by a light source. The light source casts shadows on the surface. Then the light moves to a new position where it casts new shadows. We collect the different images, of the shadowed surfaces, at these various times. From those we obtain the location of the start and the end of the shadow, plus some additional information about the surface function. Given any series of images containing shadows, we want to obtain an algorithm that produces an approximation to the surface with the smallest possible error.

We choose in this approach to recover slices of the surface. A slice is defined as the intersection of the surface and a plane of constant y (Fig. 1.) We assume that the surface slice is a function defined on the interval $[0,1]$, having continuous first derivatives, and second derivatives with bounded L_2 norm.

We propose an algorithm that recovers the surface and minimizes the worst possible error within a factor of 2. The algorithm that minimizes the error is the spline algorithm.[1] We choose to construct the spline algorithm in steps using a process that converges to the optimal error algorithm. The justification behind this stepwise construction is that, in general, the optimal error algorithm might need many iterations to construct, but in most practical cases the initial version of the algorithm, or the one resulting from a few iterations, already obtains the optimal error. We therefore construct a process that has low cost, while achieving the smallest possible error.

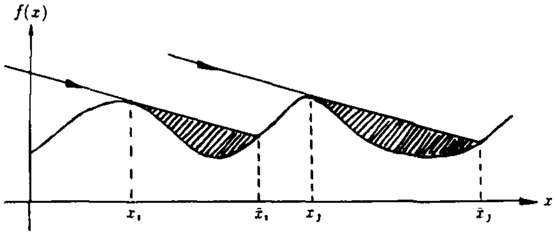


FIGURE 1.

We have done a series of numerical runs to test the performance of the above process. The obtained approximations were very close to the function from which we obtained the data, and that can be immediately seen from the pairs of initial and its reconstruction that we are supplying. We also propose a parallel implementation of the algorithm that will considerably improve the running time.

The organization of the rest of the paper will be the following : In section 2, we will formulate the problem; we will define a function space in which our surface must belong. We will also define more precisely the information that can be extracted from the shadows.

In section 3, we will define the optimal error algorithm. We show that this is the spline algorithm which always exists and is unique. This will guarantee that the shape from shadows problem under this formulation is well-posed. The definition of a well-posed problem can be found in [3, 10]. In contrast to many other *shape from X* algorithms our formulation does not require any regularization (see [8, 9] for a review of vision problems requiring regularization.)

Section 4 deals with the implementation of the optimal algorithm. Its performance is analyzed in terms of the error it creates in recovering a surface. We will show sample runs that achieve a good approximation with a small number of data.

In section 5 we discuss the cost of the proposed algorithm. We show how to take advantage of the structure of the data and modify the algorithm, so that significant cost improvements are obtained. Furthermore, the algorithm can now be implemented in parallel, resulting in an even further reduction in the running time.

2. Formulation of the problem.

In the introduction of the paper we said that our aim is to recover a 2-dimensional slice of a 3-dimensional surface. A 2-dimensional slice (see Fig. 1) can be seen as a function of one variable $f : \mathbb{R} \longrightarrow \mathbb{R}$ belonging

[1] We will define the *worst case error*, the *spline algorithm*, and all the other needed concepts later.

in the space of functions F_0. Our aim is to obtain an approximation $x \in F_0$ to our function $f \in F_0$ using the data that we can derive from the shadows. We want the approximation x to be as close to f as possible.

2.1 Function Space.

Let,

$$F_0 = \{f \mid f : [0,1] \longrightarrow \mathbb{R},\ f' \text{ absolutely cont.}, \|f''\|_{L_2} \leq 1\}, \qquad (2\text{-}1)$$

be the space that contains the functions f that we want to approximate.[2] [3] The norm $\| \cdot \|_{L_2}$ is defined as $\|f\|_{L_2} = \sqrt{\int_0^1 |f(x)|^2 dx}$.

Also, define the bilinear form $\langle \cdot, \cdot \rangle$ to be such that,

$$\langle f, g \rangle = \int_0^1 f''(x)\, g''(x)\, dx, \qquad (2\text{-}2)$$

and the norm $\| \cdot \|$ to be such that,

$$\|f\| = \langle f, f \rangle^{1/2}. \qquad (2\text{-}3)$$

Clearly $\langle \cdot, \cdot \rangle$ defined above is a semi-inner product and $\| \cdot \|$ is a seminorm. If we pose the additional requirements $f(0) = 0$ and $f'(0) = 0$ on our function, then $\langle \cdot, \cdot \rangle$ is an inner product and $\| \cdot \|$ a norm. Consequently, F_0 equipped with $\langle \cdot, \cdot \rangle$ is a semi-Hilbert space or a Hilbert space respectively.

2.2 Information.

In the next step we will extract from the image(s) the information, that is contained in the shadows, and which will be denoted by $N(f)$.[4] Clearly, from the position of the light source, we can immediately obtain the derivative of the function f at the point x_i, x_i being the beginning of the shadow (see Fig. 2). We can also obtain the difference between the two function values $f(x_i) - f(\bar{x}_i)$, at the beginning and at the end of the shadow respectively, given by $f'(x_i)(x_i - \bar{x}_i)$. Assuming that $l_i(x)$ is the straight line segment passing through the points x_i, $\bar{x}_i$, an additional piece of information can be obtained. It holds (see Fig. 2) that,

$$f(x) < l_i(x), \quad \forall x \in [x_i, \bar{x}_i]. \qquad (2\text{-}4)$$

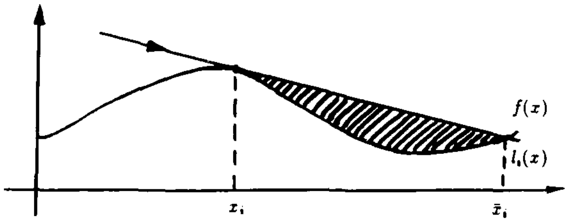


FIGURE 2.

So, formally, the information $N(f)$ contains triplets,

$$\langle f'(x_i), f(x_i) - f(\bar{x}_i), f(x) < l_i(x) \rangle. \qquad (2\text{-}5)$$

Note that the third item in the triplet is a consistency condition.

In each one of the images in our sample there are 0, 1 or more shadowed areas. From each one of those shadowed areas we can obtain a triplet of the form (2-5). If we group all the data resulting from this sampling we obtain the vector,

$$\begin{aligned} N(f) = [f'(x_1), \ldots, f'(x_n), f(x_1) - f(\bar{x}_1), \ldots,& \\ f'(x_n) - f(\bar{x}_n), f(x_1) - f(t_1), \ldots, f(x_1) - f(t_{m_1}), \ldots,& \\ f(x_2) - f(t_{m_2}), \ldots, f(x_n) - f(t_{m_n})]^{\mathsf{T}},& \qquad (2\text{-}6) \end{aligned}$$

where $m_1, \ldots, m_n$ are the number of points t_i in every interval $[x_i, \bar{x}_i]$ for which (2-4) holds, and $m_1 + \cdots + m_n = m$. Clearly, while we know that there are exactly $2n$ pieces of information in the first part of $N(f)$, we cannot bound the cardinality of the last part because, it can be the case that $m \longrightarrow +\infty$.

3. Solution of the problem - The optimal algorithm.

We now proceed to the solution of the problem. We want, given information $N(f)$ to obtain an *algorithm* φ *that will provide an approximation to our function* f. An *algorithm* is defined as any mapping from the space of all permissible data vectors to the space F_0.

3.1 Algorithm error.

The error of an algorithm for a given fixed function f is given by $\|f - \varphi(N(f))\|$. We would like to know what is the largest possible error that can be made by the algorithm, i.e. we want the error of the algorithm for the worst possible function f.

DEFINITION 3.1. *The worst case error of an algorithm* φ *is,*

$$e(\varphi, N(f)) = \sup_{\bar{f} \in F_0} \{\|\bar{f} - \varphi(N(\bar{f}))\|,\ N(\bar{f}) = N(f)\}. \qquad (3\text{-}1)$$

Clearly, we want an algorithm that minimizes $e(\varphi, N(f))$.

DEFINITION 3.2. *An algorithm* φ^* *that has the property,*

$$e(\varphi^*, N(f)) = \inf_{\varphi}\{e(\varphi, N(f))\}, \qquad \forall f \in F_0 \qquad (3\text{-}2)$$

is called a strongly optimal error algorithm.

The quantity at the right side of (3-2), i.e. the infimum of the error of all algorithms solving the problem given information $N(f)$, is a property of the problem itself, and does not depend on the particular algorithm used at any moment. This quantity, gives the inherent uncertainty of the problem for given information, and is called the *radius of information*. Clearly, the error of the strongly optimal algorithm equals the radius of *information*.[5]

3.2 The spline algorithm

We propose the spline algorithm φ^s for the solution of our problem. Splines have been known to give the optimal solution to many interesting problems [1, 2, 6, 7, 11, 12].

DEFINITION 3.3. *A spline* σ *is an element in the space of functions* F_0 *such that,*

(1) $N(\sigma) = \bar{y}$.
(2) $\|\sigma\| = \min_{f \in F_0}\{\|f\|,\ \ N(f) = \bar{y}\}$.

The meaning of (1) is that the spline must interpolate the data, and (2) says that the spline is the function that minimizes $\| \cdot \|$. The spline *algorithm is the process that constructs the spline.*

In a Hilbert space setting one can obtain a closed form for the spline algorithm. In our particular case, the spline algorithm is given by,

$$\varphi^s(x) = \sum_{i=1}^{2n} a_i g_i(x) + \sum_{j=1}^{m} c_j h_j(x). \qquad (3\text{-}3)$$

where $\{g_i\}_{i=1,\ \ ,2n}$ and $\{h_j\}_{j=1,\ \ ,m}$ are such that,

$$g_i''(x) = \frac{(x_i - x)_+^0 - (x_{i-1} - x)_+^0}{\sqrt{x_i - x_{i-1}}}, \qquad (3\text{-}4)$$

where $i = 1, \ldots, n$, $(x_i - x)_+^0 = 1$ for $x_i > x$ and 0 otherwise,

$$g_{n+i}''(x) = (x_i - x)_+ - (x_i - x)_+ - (x_i - x)_+^0(\bar{x}_i - x_i). \qquad (3\text{-}5)$$

where $i = 1, \ldots, n$, $(x_i - x)_+ = x_i - x$ for $x_i > x$ and 0 otherwise, and

$$h_j''(x) = (t_j - x)_+ - (x_i - x)_+ - (x_i - x)_+^0(t_j - x_i). \qquad (3\text{-}6)$$

[2] The bound of 1 in $\|f''\|_{L_2}$ is assumed without loss of generality. However, as we already mentioned, any fixed bound is equally good.

[3] The use of the interval $[0,1]$ is not restrictive either. Any interval $[a,b]$ for some a and b is equally good.

[4] The concept of information is considerably different from the concept of data. The data vector is a vector of fixed values, while information is an operator. We will use the term somewhat imprecisely. The user is referred to [11, 12, 13, 14] for a more detailed discussion on the concept of information operators.

[5] One point that must be mentioned here is that the radius of information describes, as we said before, the *inherent uncertainty of the problem* and has a specific value, say R. However small or large R may be, there is no procedure that will guarantee error less than that.

where $i = 1, \ldots, n$, and $j = 1, \ldots, m$.

The functions $\{g_i\}_{i=1,\ldots,2n}$ and $\{h_j\}_{j=1,\ldots,m}$ are the *representers* of the functionals that construct the information $N(f)$, properly modified to have a small area of support.

The coefficients a_i and c_j are chosen so that the definition of the spline is satisfied, that is, φ^s interpolates the data, and also minimizes the norm $\|\cdot\|$. If the area of support of every g_i is disjoint from the area of support of every other and furthermore $\langle g_i, g_j\rangle = \delta_{ij}$, the Krönecker delta, then the coefficients a_i are given directly by the theory. This is not the case in this setting where the coefficients a_i are obtained by solving a system of linear equations and the coefficients c_j are obtained by directly minimizing $\|\cdot\|$. We will define the minimization problem in section 3.3 and describe the implementation of the algorithm in section 4.

For the spline algorithm the following very strong theorem holds [7, 12].

THEOREM 3.1. *Let F_0 be a Hilbert space, $f \in F_0$ and information $\vec{y} = N(f)$. Then, the spline algorithm interpolating the data $\vec{y}$ exists, is unique, and achieves error at most twice the radius of information.*

From Theorem 3.1 we can obtain two very important results. First, our problem under the proposed formulation is well-posed. This property [3, 10] is always desirable when solving a problem. Computer vision problems tend to be ill-posed and considerable effort has been spent by the vision community towards the correct formulation that will yield well-posedness (See [4, 8, 9] for a survey.)

Second, the spline algorithm has a worst case error that is within a factor of 2 from the radius of information. The algorithm that achieves that is called *almost strongly optimal* [11]. If the problem is *linear*[6] then the spline algorithm φ^s has a worst case error equal to the radius of information and is, therefore, the strongly optimal algorithm.

The shape from shadows problem is linear, only if the cardinality of the second part of the information $N(f)$ is 0, i.e. $m = 0$. If $m > 0$ then φ^s is not strongly optimal, but almost strongly optimal as derived from Theorem 3.1.[7]

3.3 The minimization problem.

We want to mininize $\|\sigma\|^2$ where σ is given by (3-3). We can write,

$$\begin{aligned}\|\sigma\|^2 &= \langle\sigma,\sigma\rangle \\ &= \sum_{i_1=1}^{n}\sum_{i_2=1}^{n} a_{i_1}a_{i_2}\langle g_{i_1}, g_{i_2}\rangle + 2\sum_{i=1}^{n}\sum_{j=1}^{k} a_i c_j \langle g_i, h_j\rangle \\ &\qquad + \sum_{j_1=1}^{k}\sum_{j_2=1}^{k} c_{j_1}c_{j_2}\langle h_{j_1}, h_{j_2}\rangle \\ &= \vec{a}^\mathsf{T}\mathbf{G}\,\vec{a} + 2\,\vec{a}^\mathsf{T}\mathbf{P}^\mathsf{T}\vec{c} + \vec{c}^\mathsf{T}\mathbf{H}\,\vec{c}, \end{aligned} \tag{3-7}$$

where $\mathbf{G} = \{\langle g_i, g_j\rangle\}_{i,j=1,\ldots,2n}$, $\mathbf{P} = \{\langle h_j, g_i\rangle\}_{\substack{j=1,\ldots,m,\\ i=1,\ldots,2n}}$ and $\mathbf{H} = \{\langle h_i, h_j\rangle\}_{i,j=1,\ldots,m}$.

Also,

$$\begin{aligned}\langle\sigma, g_s\rangle = \sum_{i=1}^{n} a_i\langle g_i, g_s\rangle + \sum_{j=1}^{k} c_j\langle h_j, g_s\rangle = y_s \\ \Longrightarrow \qquad \mathbf{G}\,\vec{a} + \mathbf{P}\,\vec{c} = \vec{y} \\ \Longrightarrow \qquad \vec{a} = \mathbf{G}^{-1}(\vec{y} - \mathbf{P}\,\vec{c}), \end{aligned} \tag{3-8}$$

and,

$$\begin{aligned}\langle\sigma, h_s\rangle = \sum_{i=1}^{n} a_i\langle g_i, h_s\rangle + \sum_{j=1}^{k} c_j\langle h_j, h_s\rangle \le A_s \\ \Longrightarrow \qquad \mathbf{P}^\mathsf{T}\vec{a} + \mathbf{H}\,\vec{c} \le \vec{A} \\ \overset{(3\text{-}8)}{\Longrightarrow} \quad \mathbf{P}^\mathsf{T}\mathbf{G}^{-1}\vec{y} - \mathbf{P}^\mathsf{T}\mathbf{G}^{-1}\mathbf{P}\,\vec{c} + \mathbf{H}\,\vec{c} \le \vec{A} \\ \Longrightarrow \quad (\mathbf{H} - \mathbf{P}^\mathsf{T}\mathbf{G}^{-1}\mathbf{P})\,\vec{c} \le \vec{A} - \mathbf{P}^\mathsf{T}\mathbf{G}^{-1}\vec{y}. \end{aligned} \tag{3-9}$$

Now, if we substitute (3-8) for $\vec{a}$ in (3-7) we obtain,

$$\begin{aligned}\|\sigma\|^2 &= \vec{c}^\mathsf{T}\mathbf{H}\,\vec{c} + 2\,(\mathbf{G}^{-1}(\vec{y} - \mathbf{P}\vec{c}))^\mathsf{T}\,\mathbf{P}\,\vec{c} \\ &\qquad + (\mathbf{G}^{-1}(\vec{y} - \mathbf{P}\vec{c}))^\mathsf{T}\,\mathbf{G}\,(\mathbf{G}^{-1}(\vec{y} - \mathbf{P}\vec{c})) \\ &= \qquad \ldots \\ &= \vec{c}^\mathsf{T}\,(\mathbf{H} - \mathbf{P}^\mathsf{T}\mathbf{G}^{-1}\mathbf{P})\,\vec{c} + \vec{y}^\mathsf{T}\mathbf{G}^{-1}\vec{y}. \end{aligned} \tag{3-10}$$

Since $\vec{y}^\mathsf{T}\mathbf{G}^{-1}\vec{y}$ has a known fixed value for a given problem we have to minimize $\vec{c}^\mathsf{T}\,(\mathbf{H} - \mathbf{P}^\mathsf{T}\mathbf{G}^{-1}\mathbf{P})\,\vec{c}$ given the conditions in (3-9). This is a quadratic minimization problem that can be solved using a standard method.

The cardinality of the non-linear part of the information is not fixed and the choice of the value of m must be performed carefully, especially since the minimization process is costly. One may choose to always ignore the non-linear information encoded in (2-4). On the other hand non-linear information of some fixed cardinality may be always included, regardless of whether the constraints in (2-4) are violated or not.

We choose an intermediate approach which will always assure that (2-4) holds, and at the same time will minimize the cost of the algorithm (see sections 4 and 5.)

4. Application of the algorithm - Numerical runs.

The spline algorithm of section 3 has been applied and its performance has been tested in practice.

4.1 Algorithm implementation.

In our implementation the calculation of the spline algorithm proceeds in steps. From our early experience with experimental systems we have concluded that except very few cases, the non-linear part of the information is not needed. This means that the approximation produced by $\varphi^s(x)$ using information $N(f)$ with $m = 0$ does not violate the constraints (2-4).

Stage 1:

Therefore, we begin the implementation of the spline algorithm by assuming that $m = 0$ and we will first construct the values of the coefficients a_i. This is done by solving the system of equations,

$$\mathbf{G}\,\vec{a} = \vec{y}, \tag{4-1}$$

where $\mathbf{G} = \{\langle g_i, g_j\rangle\}_{i,j=1}^{2n}$ and $\{g_i\}_{i=1,\ldots,2n}$ are given by (3-4) and (3-5). The system is solved by a direct method without the need for pivoting since it is symmetric, positive definite and has a nice structure that reduces the number of calculations.

As a next step, we use the computed values of the a_i' s to construct the spline algorithm, and we plot its graph.

Third, we check to see whether the non-linear constraints are violated. This is done on line while we are plotting $\varphi^s(x)$.

If the non-linear constraints (2-4) are not violated, which as we mentioned before is usually the case, we do not need to do anything else. We have already obtained the approximation $\varphi^s(x)$ to the function f and we have plotted it. Also, since we have not used the non-linear part of the information, the problem is linear hence the spline algorithm achieves the radius of information.

Stage 2:

If the constraints (2-4) are violated, then we do not have a sufficiently good approximation, which means that we must obtain the coefficients c_j of (3-3). To do so, we have to solve the minimization problem derived in section 3.3.

We will consequently proceed as follows. We will take a few points t_i in the shadowed intervals where the constraints are violated. For these points we solve the minimization problem. Then we check again for violations of the non-linear constraints. If there are violations we repeat Stage 2. We select a few more points from the interval(s) where (2-4) is violated, and we add them to the sample. The minimization is repeated for the new set of points and the new coefficients are derived. At the same time, the a_i' s and the old c_j' s are modified.

We perform the minimization for a few points at a time for various reasons.

[6] For an exact definition of a linear problem see [7, 12, 13, 14].

[7] Strongly optimal algorithms for non-linear problems are not known in general, and if they are, they can be very difficult and expensive to calculate.

(1) It is a costly process and we would like to keep the dimensions of the problem as small as possible.
(2) At each new iteration we do not need to undo our previous work, but we simply modify the existing coefficients while deriving the new ones.
(3) We rarely need to use more that one or two points per shadowed area.

4.2 Test runs.

We have constructed a broad series of functions and we have run our algorithm on them. We started from the smoothest possible function which is a trigonometric one. Fig. 3 shows the graph of this function, and the graph of the approximating spline. A broken line is used to draw the function and a solid one is used to draw the approximating spline.

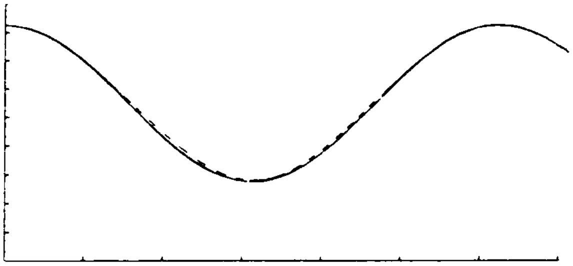


FIGURE 3.

The information we have used has been obtained for only 2 different light positions, and already yields a very close approximation. If we add samples from another 2 light positions for a total of 4, the approximation is so close to the function f that the discrepancy cannot be observed.

From the quality of this approximation we may assume that smooth functions can be approximated very well. We will move now to the other side of the spectrum which contains functions with as few derivatives as possible. This proves actually to be the most difficult case. In our setting the most irregular functions are the ones that have continuous first derivatives and discontinuous second ones. Piecewise quadratic polynomials with different second derivative from piece to piece, are functions of this type. We built many of these functions with as many as 40 different pieces each.

Additionally, in order to magnify the visual effect of any discrepancy between the function and the approximation we relaxed the assumption $||f''|| \leq 1$ that we posed in our space definition. Otherwise, the difference between f and φ^s would not be visible in any test run we would choose to show.

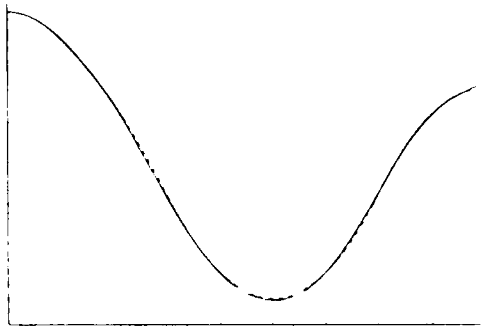


FIGURE 4.

In Figure 4 we can see the approximation to a function consisting of 10 piecewise polynomials of degree 2. The information used has been obtained from 6 different lighting angles.

It can again be seen that the function f and the approximating spline φ^s almost coincide.

We will now show one of the most difficult functions we have built, together with its approximating spline. The function consists of 40 piecewise polynomials of second degree, and has very large jumps in its second derivative, hence it has large $||f''||$. The information we have used to compute $\varphi^s(x)$, shown in Figure 5, is derived by using 8 different light angles.

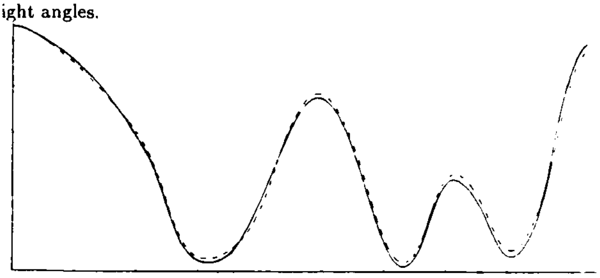


FIGURE 5.

Another issue needs to be discussed. Namely, in all the above cases the approximation to the given function has been constructed without the use of the non-linear information which, as we have mentioned, is usually the case. We will contrive a case where the use of the constraints (2-4) is needed, so that we can exhibit the second stage of the algorithm (3-3).

When,

(1) $f''(x)$ varies a lot from one polynomial piece to the other,
(2) The sampling is sparse,
(3) Both light positions are from the same side of the horizon.

Then, the approximation given by the first stage of the algorithm can fail to satisfy (2-4) (See Fig. 6.)

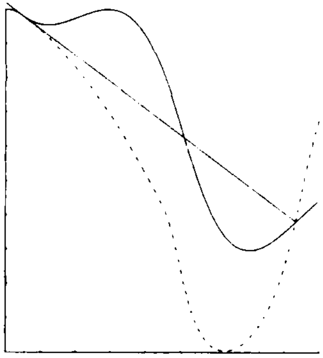


FIGURE 6.

In the example of Fig. 6 only 2 light angles were used, both from the same side of the horizon, and far apart from each other. We therefore used the second stage of the algorithm, we added two extra points t_1 and t_2 and we included $f(t_1) < l_1(t_1)$, $f(t_2) < l_1(t_2)$ in $N(f)$. Then, the constructed approximation (Fig. 7) satisfies (2-4).

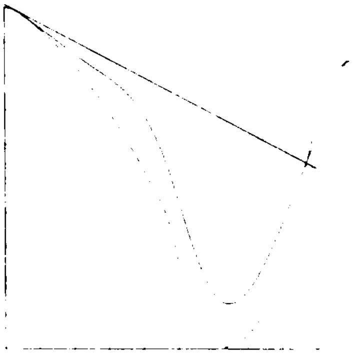


FIGURE 7.

5. Cost of the algorithm - Speed improvements.

5.1 Algorithm cost.

Let us now discuss the speed performance of our algorithm. The spline algorithm, as defined in section 3, is linear in terms of its input. Thus, if we knew the coefficients a_i and c_j then, $cost(\varphi^s)$ would be $O(n)$.

In our case, the coefficients of the spline algorithm are not known, and must be constructed. To achieve this we must solve a system of linear equations, and sometimes, a minimization problem. These costs dominate the cost of the algorithm.

In particular the solution of the system (4-1) has a cost $O(n^3)$. The cost of the quadratic minimization is considerably higher, that is it is exponential in terms of the number of non-linear information samples used in each stage. In this case, and since we have observed that better and denser sampling alleviates the need to use the second stage of the algorithm, we might choose to increase the cardinality of the sampling, if that can be done, and solve a slightly larger linear problem instead.

5.2 Speed improvements.

In section 5.1 we have discussed the cost of a very straightforward implementation of the spline algorithm described in section 4.1. We now show that a slight improvement in the implementation of the algorithm can yield a significant speedup. This speedup can be achieved only if the function we want to recover can be split in distinct sections that we will from now on call *valleys*. A valley is defined by two local maxima of the function, but also depends on the specific sampling. For example, the function of Fig. 1 has 2 valleys. In particular, we say that the function f, under some fixed sampling, has k valleys if we can define k partitions $\Pi_1, \Pi_2, \ldots, \Pi_k$ of the functions $\{g_i\}_{i=1,\ldots,2n}$, given by (3-4) and (3-5), such that the union of the areas of support of all the functions in each partition is disjoint from the union of the areas of support of the functions in every other partition.

We can detect the existence of any number of valleys in time $O(n)$ and subsequently, we can solve k problems of sizes $n_1, n_2, \ldots, n_k$ respectively, instead of solving one problem of size n, where $n = n_1 + n_2 + \cdots + n_k$.

To connect the pieces resulting from each of the k problems we need constant time per problem, hence combining can be done in time $O(k)$.

Therefore, the total cost of this algorithm, which we will denote φ_i^s, will be $O(k\nu^3)$, where $\nu = \max\{n_1, \ldots, n_k\}$.

5.3 Parallel implementation.

Since splitting the problem into individual subproblems and combining the resulting surfaces is straightforward and cheap to implement, one immediate extension to the above set-up of the problem is to assign one individual subproblem to a different processor and solve the initial problem in a parallel or in a distributed environment.

Again, splitting into k subproblems requires time $O(n)$ and combining the individual solutions into one requires time of $O(k)$. Then every processor will require time $O(n_i^3)$, $i = 1, \ldots, k$ resulting in a total cost for the parallel version φ_p^s of our algorithm of $O(\nu^3)$, where $\nu = \max\{n_1, \ldots, n_k\}$.

Let us now compare the three different implementations of the spline algorithm, using a specific example. Assume we have information $N(f)$ of cardinality $n = 512$.[8] Also assume that the number of valleys k is 8. We assume without loss of generality that $n_1 = n_2 = \cdots = n_k = \nu = 64$. For these values of n, ν, and k the performance figures listed in Table 1 are obtained.

Algorithm	Cost (Millions of ops)
φ^s	$512^3 \approx 134.4$
φ_i^s	$8 \cdot 64^3 \approx 2.0$
φ_p^s	$64^3 \approx 0.26$

TABLE 1.

It is apparent from the above table that we can obtain substantial speed improvements with very low added overhead. It should additionally be noted that we can obtain a good approximation using around 8 to 10 different light positions. If this is the case, the only occasion when we can obtain $N(f)$ of cardinality $n = 512$ is if k is very large, i.e. if we have a big number of valleys. In that case the speed improvements should be even larger than the ones already exhibited. Conversely, if the number of valleys is small then the size of n is expected to be low, since it is proportional to the number of light positions, and will in no case reach the magnitude of the above example.

6. Conclusion - Future work.

We solved the problem of recovering a one-dimensional surface slice from the shadows it casts on itself when lighted by a light source positioned at various locations.

We proposed a formulation that results in a well-posed problem and we have consequently proceeded into solving it. We proposed an optimal error algorithm which additionally achieves a low time cost, especially if a clever but simple breakdown of the problem is used.

There are many aspects of this problem that can be looked at in the future. A method for the faster solution of the optimization problem, based on its specific structure, is one of them. On the other hand it would be useful to quantify whether the use of the non-linear information can be avoided. Our current belief is that good sampling can take care of all cases. The second stage of the algorithm might be still useful in cases where we have to deal with a fixed, given, not very appropriate sampling.

Another natural extension to the shape from shadows problem is to try to recover the whole 3-D surface instead of recovering surface slices, as we are doing in this paper. We are currently working on this interesting aspect.

7. Acknowledgments.

M. Hatzitheodorou would like to thank Prof. Greg Wasilkowski, currently with the University of Kentucky at Lexington, for the fruitful discussions and comments he offered during the starting stages of this work.

REFERENCES

[1] Anselone, P. M., and Laurent, P. J., *A general method for the construction of interpolating or smoothing spline functions*, Nummer. Math. **12** (1968).

[2] Atteia, M., *Fonctions spline généralisées*, C.R. Acad. Sci. Paris **261** (1965).

[3] Hadamard, J., *Sur les problèmes aux derivées partielles et leur signification physique*, Princeton Univ. Bulletin **13** (1902).

[4] Hatzitheodorou, M. G., *The Application of Approximation theory methods to the solution of computer vision problems*, (In Progress).

[5] Kender, J. R., and Smith, E. M., *Shape from darkness. Deriving surface information from dynamic shadows*, Proceedings AAAI (1986).

[6] Holmes, R., *R-splines in Banach spaces : I. Interpolation of linear manifolds*, J. Math. Anal. Appl. **40** (1972).

[7] Michelli, C. A., and Rivlin, T. J, *A Survey of optimal recovery*, in "Optimal estimation in Approximation theory," Plenum Press, 1977.

[8] Poggio, T., *Computer vision*, MIT Artificial Inteligence Lab (1986).

[9] Poggio, T., and Torre, V., *Ill-posed problems and regularization analysis in early vision*, MIT Artificial Inteligence Lab (1984).

[10] Tikhonov, A. N., and Arsenin, V. Y., "Solutions of ill-posed problems," *V.H.Winston* and Sons, 1977.

[11] Traub, J. F., Wasilkowski, G., and Woźniakowski, H., "Information, Uncertainty, Complexity," Addison-Wesley, 1983.

[12] Traub, J. F., and Woźniakowski, H., "A general theory of optimal algorithms," Academic Press, 1981.

[13] Woźniakowski, H., *A survey of information-based complexity*, Journal of Complexity **1** (1985).

[14] Woźniakowski, H., *Information-based complexity*, Annual Review of Computer Science **1** (1986).

[8] A very high number compared to the values of n used in the sample runs we have shown.

PREDICTING MATERIAL CLASSES

Glenn Healey and Thomas O. Binford

Robotics Laboratory
Computer Science Department
Stanford University
Stanford, CA 94305

Abstract

Given the physical properties of a class of materials, it is possible to predict the properties of images of these materials. In this paper, we examine the properties of metals and dielectrics which allow them to be distinguished in color images. We begin by adopting general physical models for the properties of materials which determine how they interact with light. From these models, we analyze the geometric, spectral, and intensity properties of the light reflected from an arbitrary material. The goal of this work is to isolate those properties of the reflected light which provide evidence for the classification of a material as either a metal or a dielectric. The ability to achieve this level of classification is a first step towards a more general material identification system.

1 Introduction

When an electromagnetic wave is incident on the surface of an object, it will interact with the material of the object. The nature of this interaction depends on the electrical and magnetic properties of the object's material. For the special case of visible light, the relevant properties are called optical properties. In this work, we develop methods to predict information about images of a material from the optical properties of the material.

An important electrical property of a material is the degree to which it conducts electricity. Metals are good conductors of electricity, while dielectrics are not. This fundamental difference causes metals and dielectrics to interact differently with light. If we can understand how this difference in interaction causes corresponding differences in images of these classes of materials, we can derive methods to distinguish metals from dielectrics in images.

There are several reasons why it is useful in image understanding to be able to distinguish metals from dielectrics. Certainly, material identification is an important part of building descriptions of objects in a scene. In object recognition tasks, material identification provides a valuable cue for recognition. Aside from this general utility, the ability to distinguish metals from dielectrics is useful for another reason. Since metals and dielectrics interact with light in fundamentally different ways, algorithms have been developed in computer vision which apply to one of these classes of materials but not to the other. For example, certain vision algorithms require the identification of a dielectric material in a scene [4], [7]. Other algorithms are better suited for metals, e.g. [10]. Before these algorithms can be applied to images, it is necessary to determine if a material in the scene is a metal or a dielectric.

2 Preliminaries

In this section, we give an overview of the interaction of light with matter and introduce much of the terminology which will be used in the rest of this work.

2.1 Optically Homogeneous and Inhomogeneous Materials

It is useful to divide materials into two classes based on their optical properties. This subsection defines optically homogeneous and optically inhomogeneous materials. The methods described in this work apply to both kinds of materials.

Optically homogeneous materials have a constant index of refraction throughout the material. Consequently, for an object composed of a homogeneous material in air, light undergoes reflection and refraction only as it encounters a surface of the object. Metals and crystals are the most common examples of homogeneous materials.

Optically inhomogeneous materials are composed of a vehicle with many embedded colorant particles which differ optically from the vehicle. While in the body

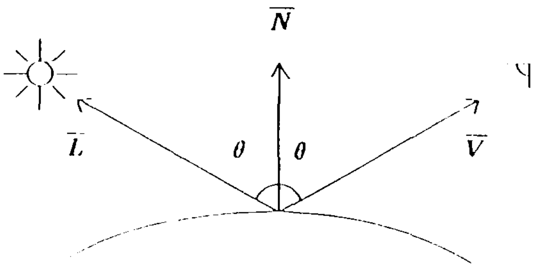


Figure 1. Specular Reflection

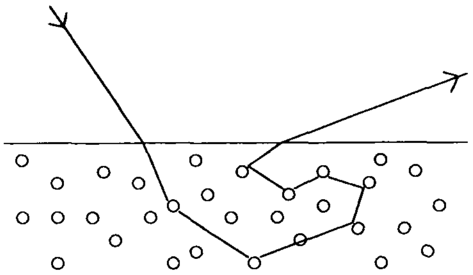


Figure 2. Colorant Layer Scattering

of an inhomogeneous material, light typically interacts with many colorant particles. The net result of many interactions is that the light is diffused. Some examples of inhomogeneous materials are plastics, paper, textiles, and paints.

2.2 Reflection

When light is incident on the surface of an object, some fraction of it is specularly reflected. A smooth surface reflects light only in the direction such that the angle of incidence equals the angle of reflection (Figure 1). The properties of this specularly reflected light are determined by the optical and geometric properties of the surface. For optically homogeneous materials which are not transparent, appearance is completely determined by the properties of specularly reflected light. For many inhomogeneous materials, such as plastics, specular effects are also significant.

The fraction of the incident light which is not specularly reflected enters the body of the material. For inhomogeneous materials, the body is composed of a vehicle and many colorant particles. When light encounters a colorant particle, some portion of it is reflected. After many reflections, the light is diffused and a significant fraction can exit back through the surface in a wide range of directions (Figure 2).

2.3 Irradiance and Color

In computer vision, imaging sensors are typically used to measure certain properties of light across a plane. Two of the properties of greatest interest in computer vision are irradiance and color. Irradiance is a single measure of how much visible light strikes an area of a sensor plane. It is defined as power per unit area. Irradiance, therefore, contains no direct information about the spectral properties of the light striking the sensor plane, i.e., many different spectral power distributions will give the same image irradiance. We define the color of light striking a sensor plane to be the light's spectral power distribution in the visible range. For an image of a reflecting surface, image irradiance and color will depend on the properties of the light illuminating the surface, the spectral reflectance of the surface material, and the geometry of reflection. A detailed discussion of the sensing and representation of irradiance and color in computer vision is given in [3].

2.4 The Reflectance Model

We quantify the reflectance R of a surface by

$$R(\theta_l,\theta_v,\theta_p,\lambda) = R_S(\theta_l,\theta_v,\theta_p,\lambda) + R_B(\theta_l,\theta_v,\theta_p,\lambda) \quad (1)$$

where R_S is the specular reflectance term and R_B is the body reflectance term due to colorant layer scattering. The angles θ_l, θ_v, and θ_p are the photometric angles. θ_l denotes the angle between the surface normal and the illumination direction. θ_v denotes the angle between the surface normal and the viewer. θ_p denotes the angle between the illumination direction and the viewing direction. As usual, λ denotes wavelength.

The power of the light reflected towards a viewer is given by

$$I(\theta_l,\theta_v,\theta_p,\lambda) = R(\theta_l,\theta_v,\theta_p,\lambda)L(\lambda) \quad (2)$$

where $L(\lambda)$ is the spectral power distribution of the light incident on the surface.

Table 1. Electromagnetic Properties of Matter

Property	Symbol	Units
Electrical permittivity	ϵ	$C^2/(N \cdot m^2)$
Magnetic permeability	μ	$N \cdot s^2/C^2$
Conductivity	σ	$(\Omega \cdot m)^{-1}$

3 The Optical Properties of Matter

In this work, we treat light as an electromagnetic wave. While at a fundamental level certain properties of light are best described in terms of localized quanta of energy, the light we measure in this work transports relatively large amounts of energy. Hence the effects of individual quanta cannot be distinguished. For this work, electromagnetic theory is quite adequate for describing the propagation of light.

Given our view of light as an electromagnetic wave, the characteristics of the interaction between light and material depend on the material's electrical and magnetic properties. Those material properties which appear in Maxwell's equations and thereby determine the interaction are summarized by the quantities ϵ, μ, σ (Table 1).

The magnetic permeability of all materials, with the exception of ferromagnetic materials, is very nearly equal to the permeability of free space μ_0. Since ferromagnetic materials are scarce, we do not consider them in this work. Consequently, we take $\mu = \mu_0$.

It is important to note that ϵ and σ are not constants for a given material, but in fact depend on the wavelength of the light being considered. This dependence on wavelength will prove to be important when we consider the spectral properties of the light reflected from a material.

The optical properties of a material are summarized by the complex index of refraction $M = n - iK_0$ where both n and K_0 are real. The dimensionless quantity n is called the refractive part of M. In a nonattenuating medium ($K_0 = 0$), n is defined as the ratio of the speed of light in a vacuum to the speed of light in the material. For absorbing media ($K_0 > 0$), the interpretation of n is more complicated. K_0 is called the absorptive part of M or the extinction coefficient. If the amplitude of an electromagnetic wave in a material does not decrease as the wave propagates in the material, then $K_0 = 0$. In a homogeneous absorbing material, the amplitude of a light wave is exponentially attenuated. If a plane wave of angular frequency w traveling in the x direction encounters an interface at $x = 0$, then the irradiance $I(x)$ at a depth x in the material is given by

$$I(x) = I(0)e^{-2wK_0x/c} \tag{3}$$

where c is the speed of light in a vacuum. The distance $x = c/(2wK_0)$ is often called the skin depth for a material. From equation 3, we see that the irradiance decreases by a factor of e^{-1} as the light wave penetrates to the skin depth.

It is possible to solve for n and K_0 in terms of the fundamental electromagnetic properties of the material by requiring that the incident light wave satisfies Maxwell's equations. The resulting expressions for n and K_0 are given by

$$n^2 = \frac{\mu_0 \epsilon c^2}{2}\left[1 + \left[1 + \left(\frac{\lambda\sigma}{2\pi c\epsilon}\right)^2\right]^{\frac{1}{2}}\right] \tag{4}$$

$$K_0^2 = \frac{\mu_0 \epsilon c^2}{2}\left[-1 + \left[1 + \left(\frac{\lambda\sigma}{2\pi c\epsilon}\right)^2\right]^{\frac{1}{2}}\right] \tag{5}$$

Since ϵ and σ are, in general, functions of wavelength, both n and K_0 are therefore functions of wavelength.

Given isotropic homogeneous media, the Fresnel equations describe the light which is specularly reflected from an interface in terms of M and geometry. If unpolarized light is incident at an angle θ_i, then the monochromatic specular reflectance R_S is

$$R_S = 0.5(R_\perp + R_\parallel) \tag{6}$$

where $R_\perp$ is the perpendicular polarized component and $R_\parallel$ is the parallel polarized component. From electromagnetic theory, $R_\perp$ and $R_\parallel$ are given by

$$R_\perp = \frac{a^2 + b^2 - 2acos\theta_i + cos^2\theta_i}{a^2 + b^2 + 2acos\theta_i + cos^2\theta_i} \tag{7}$$

$$R_\parallel = R_\perp \frac{a^2 + b^2 - 2asin\theta_i tan\theta_i + sin^2\theta_i tan^2\theta_i}{a^2 + b^2 + 2asin\theta_i tan\theta_i + sin^2\theta_i tan^2\theta_i} \tag{8}$$

where

$$a = 0.5\sqrt{\sqrt{g^2 + 4n^2K_0^2} + g} \tag{9}$$

$$b = 0.5\sqrt{\sqrt{g^2 + 4n^2K_0^2} - g} \tag{10}$$

$$g = n^2 - K_0^2 - sin^2\theta_i \tag{11}$$

Equations 7 and 8 are known as the Fresnel equations. The Fresnel equations are derived in many places including [1].

For the case of normal incidence, i.e. $\theta_i = 0°$, equation (6) simplifies to

$$R_S(\theta_i = 0°) = \frac{(n-1)^2 + K_0^2}{(n+1)^2 + K_0^2} \tag{12}$$

Table 2. Conductivity of Materials

Material	Class	Conductivity
copper	metal	5.9×10^7
zinc	metal	1.7×10^7
iron	metal	1.0×10^7
uranium	metal	3.4×10^6
graphite	semi-conductor	2.0×10^4
germanium	semi-conductor	$10^1 - 10^3$
silicon	semi-conductor	$10^{-1} - 10^1$
glass	ceramic	$10^{-11} - 10^{-9}$
diamond	ceramic	$10^{-11} - 10^{-10}$
porcelain	ceramic	$10^{-12} - 10^{-10}$
mica	ceramic	$10^{-11} - 10^{-15}$
fused quartz	ceramic	$10^{-16} - 10^{-18}$

4 The Interaction of Light with Metals

In this section, we examine the characteristics of the interaction of light with metals. The most prominent difference between metals and other materials is the large number of electrons which are free to move throughout a metal. Metals, therefore, have large values of the conductivity σ. By contrast, electric charges are not free to move in dielectrics, making these materials poor conductors. Table 2 shows the large variation of conductivity among various classes of materials.

Another important property of metals is that they are optically homogeneous. As a result, colorant layer scattering does not occur for metals and $R_B \equiv 0$ in equation (1). Therefore, the only relevant reflection process for metals is specular reflection.

In subsections 4.1 and 4.2, we consider respectively the intensity and spectral properties of the light reflected from a metal.

4.1 Intensity Properties of the Reflected Light

The conductivity σ of a material is defined as the ratio of the current density produced by an electric field to the intensity of the electric field. In a perfect dielectric, there are no free electrons and the conductivity is zero. For metals, on the other hand, there are many free electrons and the conductivity is much greater than zero (Table 2). In the hypothetical case of a perfect conductor, we would have $\sigma = \infty$. Physically this would mean that a stream of electrons moving in a certain direction would continue to move in that direction without resistance. In real metals, however, electrons experience resistance due to departures of the solid lattice from perfect regularity. These departures from perfect regularity are due to thermal motion of the atoms and impurities in the solid. When an electron is scattered by the imperfect lattice, electromagnetic energy is irreversibly converted to heat. This process is in part responsible for the attenuation of the incident wave as described by equation 3.

We begin by approximating metals as perfect conductors, i.e. we let $\sigma \rightarrow \infty$. If we fix λ to be some visible wavelength, then n and K_0 are functions of only ϵ and σ. It is not possible to directly measure ϵ for metals. We can assume from the relevant physics, however, that ϵ for metals is the same order of magnitude as for dielectrics [1]. Therefore, from equation 5, we have $K_0 \rightarrow \infty$ as $\sigma \rightarrow \infty$. From the Fresnel equations, this tells us that $R_S \rightarrow 1$ as $\sigma \rightarrow \infty$.

Although the conductivity of real metals is finite, the approximation $R_S(\theta_i, \lambda) \approx 1$ is reasonable for many metals in the visible wavelength range. At the physical level, this is because the many free electrons present in a metal are excellent scatterers of light. Thus incident light has little opportunity to penetrate into the metal. Consequently, K_0 is large and metals have a skin depth which is only a small fraction of a wavelength. Few electrons are able to absorb any energy from the incident light, causing most of the incident light to be reflected.

4.2 Spectral Properties of the Reflected Light

In this section we investigate certain physical properties of metals and explain how these properties determine the structure of a metal's surface spectral reflectance function. We show from first principles that the surface spectral reflectance functions for metals are highly constrained. This is of great use in material identification since it allows a system to readily decide from a material's estimated spectral reflectance if that material could possibly be a metal.

The optical properties of a metal are largely determined by how light interacts with the electrons in the metal. We can divide the various possible interactions into two classes. In the first class of interaction, light is scattered by a free conduction electron. This class of interaction is easily the most common in metals because of their large number of free electrons. In the second class of interaction, the light's energy is absorbed by the metal as an electron jumps to a higher energy level. We examine in more detail the properties of these two classes of interaction in the following paragraphs.

The most prominent process which contributes to the optical properties of a metal is the interaction of light with free conduction electrons. As explained in 4.1, these free electrons scatter light very effectively causing metals to have a small skin depth and large reflectance values. Since these free electrons have no natural frequencies, they scatter light of all wavelengths equally well. This is why most metals (e.g. aluminum, silver, sodium) have spectral reflectance functions which do not vary with wavelength in the visible range and thus appear silver or gray in color.

In addition to being scattered by free electrons, light can also be absorbed during interactions with electrons which are confined to exist in certain energy zones (called Brillouin zones). Each zone is separated from adjacent zones by a gap of disallowed energies. For a certain range of incident light wavelengths, the energy of the incident light will match the energy gap between zones. In this situation, an atom will absorb the light allowing an electron to jump to a higher energy zone. This absorbed energy is typically converted rapidly into heat. The important point is that there is a critical wavelength λ_0 such that incident light energy is not absorbed for $\lambda > \lambda_0$ and for decreasing λ there is a dramatic increase in the amount of incident light energy absorbed at λ_0. This absorbed light energy cannot appear in the reflected wave. For decreasing wavelength, therefore, n and K_0 behave in such a way that there is a dramatic decrease in spectral reflectance at λ_0.

For most metals, there is no absorption of light in the visible wavelength range. Thus most metals have spectral reflectance functions which are constant with respect to wavelength in the visible. The most notable exceptions to this rule are copper and gold, both of which have absorption bands in the visible [9]. The spectral reflectance functions for both copper and gold are near one for long visible wavelengths and decrease sharply below their respective critical wavelengths λ_0. Figure 3 shows the spectral reflectance functions for copper and gold. Copper and gold strongly absorb blue and green light while reflecting red light. This gives these metals their reddish colors. Other metals have absorption bands which do not lie in the visible. Silver, for example, begins absorbing in the ultra-violet at about $\lambda_0 = 310$nm.

We have shown that the physical properties of metals provide strong constraints on their spectral reflectance functions. The scattering of light by free conduction electrons causes most metals to have spectral reflectance functions which are constant over the visible wavelengths (e.g. silver, aluminum, sodium, tin, potassium, niobium, scandium, cesium, yttrium, vanadium, niobium, osmium). For other metals, absorption bands in the visible give rise to critical wavelengths λ_0 such that the metal reflects strongly for $\lambda > \lambda_0$ and absorbs strongly for $\lambda < \lambda_0$ (e.g. copper, gold). Note that these are the only possible kinds of spectral reflectance functions for a metal. We would not expect, for example, to see metals with spectral reflectance functions that decrease sharply with increasing wavelength. This is why there are no metals that appear green or blue. Consequently, it is often easy to decide from a spectral reflectance function that a material is not a metal.

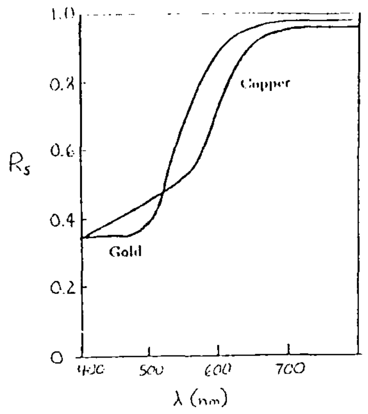


Figure 3. Spectral Reflectance of Copper and Gold

5 Light and Dielectrics

In this section, we examine the characteristics of the interaction of light with dielectric materials. The most significant difference between dielectrics and metals is that dielectrics have very small conductivities compared to metals (Table 2). Physically this occurs because in a dielectric all Brillouin zones which contain any electrons are full and there exist large energy gaps separating these full zones from adjacent empty zones. Hence an applied electric field does not produce a current and the material is an insulator.

Another important difference between dielectrics and metals is that many dielectric materials are optically inhomogeneous. Therefore, for most dielectrics both specular reflection and colorant layer scattering are important optical processes.

5.1 Homogeneous Dielectrics

In this subsection, we consider the properties of a homogeneous material which is a perfect dielectric. For a perfect dielectric, the conductivity σ is zero. Therefore, from (4) and (5) we have

$$n = \sqrt{\mu_0 \epsilon c^2} \tag{13}$$

$$K_0 = 0 \tag{14}$$

For homogeneous dielectrics, n defined by (13) is equal to the ratio of the speed of light in a vacuum to the speed of light in the material. Since K_0 is zero in (14), we see that a homogeneous dielectric medium does not attenuate the incident light. The normal incidence Fresnel reflectance for such a material is

$$R_S(\theta_I = 0^\circ) = \frac{(n-1)^2}{(n+1)^2} \tag{15}$$

For the values of n corresponding to most dielectrics, R_S is small. In particular, R_S for metals is typically much larger than R_S for dielectrics.

5.2 Inhomogeneous Dielectrics

Most dielectric materials, with the exception of crystals, are optically inhomogeneous. Some examples of inhomogeneous dielectric materials are plastics, paints, textiles, and ceramics. As explained in 2.2, the body of an inhomogeneous material consists of a vehicle and many embedded colorant particles. The refractive index n of the vehicle determines the specular reflectance of the material from the Fresnel relations (e.g. as in equation 15). For most vehicle materials, n is only weakly dependent on λ [12]. The fraction of the incident light which enters the body of the inhomogeneous dielectric interacts with many colorant particles. These colorant particles selectively absorb certain wavelengths and thereby change the spectral properties of the incident light. The colorant particles also serve to diffuse the incident light and much of it is scattered out of the body of the material in many directions. The modified Kubelka-Munk theory [8], [11] provides a model for the scattering of light from inhomogeneous dielectrics. The use of this model is discussed in [5].

6 Identifying Metals and Dielectrics

In sections 4 and 5, we showed that there are significant differences in the ways in which metals and dielectrics interact with light. In this section, we summarize these differences. We are currently developing a computational technique based on these differences and geometrical considerations which can be used to identify metals and dielectrics in images.

In section 4 we showed that metals reflect almost all the incident light in nearly a single direction. We also showed that the spectral reflectance functions for metals are highly constrained. There is no colorant layer scattering for metals. Thus, for a single spectral power distribution of illumination, the color of light reflected from a metal will be nearly constant with only slight changes due to changing geometry. Measurements are given in [2] demonstrating this property.

In section 5, we considered the optical properties of dielectrics. For inhomogeneous dielectrics, colorant layer scattering is an important optical process. Dielectrics typically have a small specular reflectance component. This specular reflectance component usually depends only weakly on wavelength. The light scattered from the body of a dielectric is distributed over a wide range of angles and is generally of a different color than the specularly reflected light. Measurements depicting this color shift are given in [4] and [7]. Another distinguishing property of dielectrics is that their body spectral reflectance functions can take on much more general forms than the spectral reflectance functions for metals.

In this work, we have considered primarily clean, smooth surfaces. We have not taken into account the complicating effects of surface roughness and surface impurities (e.g. oxides on metals) which can affect the properties of the light scattered by a surface. The effects of both roughness and impurities are examined in [6]. We believe, however, that the fundamental nature of the differences between metals and dielectrics will allow them to be distinguished in images despite the effects of roughness and impurities.

Acknowledgements

Support for this work was provided in part by the Air Force Office of Scientific Research under contract F33615-85-C-5106 and by the Advanced Research Projects Agency of the Department of Defense under Knowledge Based Vision ADS subcontract S1093 S-1.

8 References

1. Born, M. and Wolf, E., *Principles of Optics*, Pergamon Press, New York, 1959.

2. Healey, G. and Binford, T.O., "The Role and Use of Color in a General Vision System," submitted to *International Journal of Computer Vision*.

3. Healey, G. and Binford, T.O., "A Color Metric for Computer Vision," Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, June 1988. Also appears in Proceedings of ARPA Image Understanding Workshop, April 1988.

4. Healey, G. and Binford, T.O., "Color Algorithms for a General Vision System," *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, Milan, Italy, pp. 759-762, August 1987.

5. Healey, G. and Binford, T.O., "The Role and Use of Color in a General Vision System," Proceedings of ARPA Image Understanding Workshop, USC, 1987.

6. Healey, G. and Blanz, W.E., "Identifying Metal Surfaces in Color Images," SPIE 1988 Technical Symposium on Optics, Electro-Optics, and Sensors, Orlando, April 1988.

7. Klinker, G. and Shafer, S. and Kanade, T., "Using a Color Reflection Model to Separate Highlights from Object Color," *Proceedings of the First International Conference on Computer Vision*, London, pp. 145-150, June 1987.

8. Kubelka, P. and Munk, F., "Ein Beitrag sur Optik der Farbanstriche," Z. tech. Physik., 12, 1931, 593.

9. Mott, N. and Jones, H., *The Theory of The Properties of Metals and Alloys*, Oxford University Press, 1958.

10. Porter, G. and Mundy, J., "Automatic Visual Inspection of Metal Surfaces," *Proceedings of SPIE, Vol. 281, Techniques and Applications of Image Understanding*, pp. 176-181, 1981.

11. Reichman, J., "Determination of Absorption and Scattering Coefficients for Nonhomogeneous Media. 1: Theory," *Applied Optics*, Vol. 12, No. 8, August 1973, 1811-1815.

12. Siegal, R. and Howell, J., *Thermal Radiation Heat Transfer*, McGraw-Hill, 1981.

Texture Edge Localization

Richard Vistnes

Robotics Laboratory, Stanford University
Stanford, California 94305

Abstract

Texture segmentation may be separated into two phases: edge detection, followed by edge localization; this paper focuses on the latter topic. In the first part of this paper, we use some simple, abstracted textures to illustrate the issues involved in texture edge localization. We discuss some of the limits of localization. The second part suggests a simple approach to localization based on interpreting the responses of edge detectors spaced at fixed intervals in position and orientation space. The results of this algorithm as applied to some synthetic and natural textures are presented.

1 Introduction

Texture segmentation seeks to locate image curves that separate different textures. The segmentation problem may be separated into two phases: detection of texture differences between regions, and localization of the edges thus detected. The current paper focuses on the latter subject, edge localization.

We briefly review the current approach to texture analysis; the reader is encouraged to refer to [10] and [11] for a more complete discussion. We define textures by statistical distributions, and define texture edges by differences in distributions between two regions. Edge detection amounts to estimating distributions over two regions and comparing the estimated parameters. We use the statistical significance of the difference between parameters as a measure of the difference between two textures. As a simple example, if a texture is composed of pixels with normally-distributed intensity, the distribution is defined by its mean and variance, which may be estimated over a region. Statistical techniques can be used to compute the probability of a mean/variance difference as large as observed, under the null hypothesis that the underlying distributions are identical in the two regions.

Texture discrimination requires making assumptions about the nature of the distributions from which the textures arise. If the domain is controlled (e.g., a factory), assumptions can be based on scene knowledge. When faced with images of unknown scenes, one must make general assumptions that allow one to discriminate a large number of different textures. From such assumptions, one derives a set of texture features to measure from the image; discrimination is based on comparisons of such texture measures, and the textures discriminable by a vision system depend on these measures. A set of texture measures that allows reliable discrimination of natural textures is discussed in [11]; the demonstrations discussed below use those measures.

The current approach to texture segmentation is based on computing the "strength" of a texture edge at many positions in the image, and then interpreting the resulting array of edge strengths to localize the edges. Edge operators that compute the statistical significance of a texture boundary for a hypothesized edge between two regions are discussed at length in [10]. Generally, the maximum edge significance occurs when the hypothesized edge coincides with the underlying edge. However, the edge significance is high when the hypothesized edge is *near* the underlying edge as well. The problem of edge localization is to identify the image locations that provide the best estimate of the underlying edges.

Edge localization is a difficult problem. In simple images, edges may be well defined and easily localized, and in these cases it is reasonable to design perceptually valid localization algorithms. For example, the edges in a blocks-world scene are typically well defined and can easily be identified by a human observer. However, in more complex scenes, such as natural scenes, the dominant edges may be very difficult to identify. The *scale problem* is particularly difficult: edges exist at many scales, and the appropriate scale of analysis must be determined by the application. A natural scene may contain boundaries between a forest and the sky, between individual trees in the forest, and edges around the leaves within the trees; depending on the application, one or more scales of edge may be appropriate.

In this paper, we first examine some simple, abstracted texture images that illustrate many of the issues involved in edge localization. We discuss the limits of localization in such textures. The remainder of the paper focuses on one simple approach to localization. This approach is based on computing edge significances at many locations and orientations in the image, and finding patterns in the resulting edge significance arrays that correspond to image texture edges. The edge localization algorithm is demonstrated on a natural textured image

2 Edge Localization Limits

Texture edge localization is limited by the information present in the image. Estimates of texture edge location are based on comparisons of estimated parameters of texture distributions. In one important type of texture, the relevant texture features are attributes (such as length or orientation) of image structures. Since these image structures are sparse in the image, texture information is sparse: where there are no structures, there is no information. As a result, localization accuracy increases with the density of image structures.

For the purpose of exposition in the following discussion, let us consider the abstracted texture image depicted in Figure 1a. This texture consists of a two-dimensional array of uniformly-placed elements with one attribute; the attribute may be brightness, orientation, etc., but in the figure, it is represented as size. (In the case of general textures, the elements have many attributes rather than just one, and the elements are not well delineated in the image.) The array is divided into two regions, the left hand side (LHS) and right hand side (RHS)[1]; the element attributes on either side are chosen from different normal distributions. These 2D, uniform textures can be characterized by their means and variances: μ_L and σ_L^2 on the LHS, and μ_R and σ_R^2 on the RHS. We shall say that the textures differ when their means differ, i.e., when $\mu_L \neq \mu_R$.

In this highly simplified domain, texture segmentation involves locating the curve that separates the two regions. Imagine a curve C that passes through the texture from top to bottom. For each such curve we can measure the sample means ($\overline{x}_L$ and $\overline{x}_R$) and variances (σ_L^2 and σ_R^2) on either side of C, and compute the statistical significance of the edge across C by testing the hypothesis $H_0 : \mu_L = \mu_R$. More precisely, a significance test computes the probability $P(D \mid H_0)$, which decreases as D, the observed difference in sample means, increases. It is more natural to use as a difference measure the significance of the difference, $\Delta(C) = -\log P(D \mid H_0)$, which is always positive and grows with the significance of the edge[2].

In image segmentation, we generally wish to find the "simplest" curve through the image, choosing a linear boundary if possible. Suppose we attach to a curve C a measure of simplicity $S(C)$ that is large when C is linear and decreases as C becomes more curved or jagged. (One ad hoc measure of curve simplicity is the ratio of endpoint-endpoint length to total arc length; this is unity for lines and decreases with the jaggedness or curviness of the curve.) Our criteria for the choice of a boundary curve C depend on some combination of its simplicity $S(C)$ and the significance $\Delta(C)$ of the edge across it. The edge localization problem amounts to maximizing this combined quality measure over all C. In general, several curves may satisfy the "goodness" criteria equally well; this is one source of uncertainty in localization. In practice, of course, it is impossible to test all curves passing through the image; they must be constrained in some way. One solution is to consider only piecewise linear curves, or 'edgels', at fixed positions and orientations in the image. Since linear C have identical simplicity $S(C)$, the problem of measuring curve simplicity is largely avoided. Measuring $S(C)$ in a well-motivated manner remains an open problem.

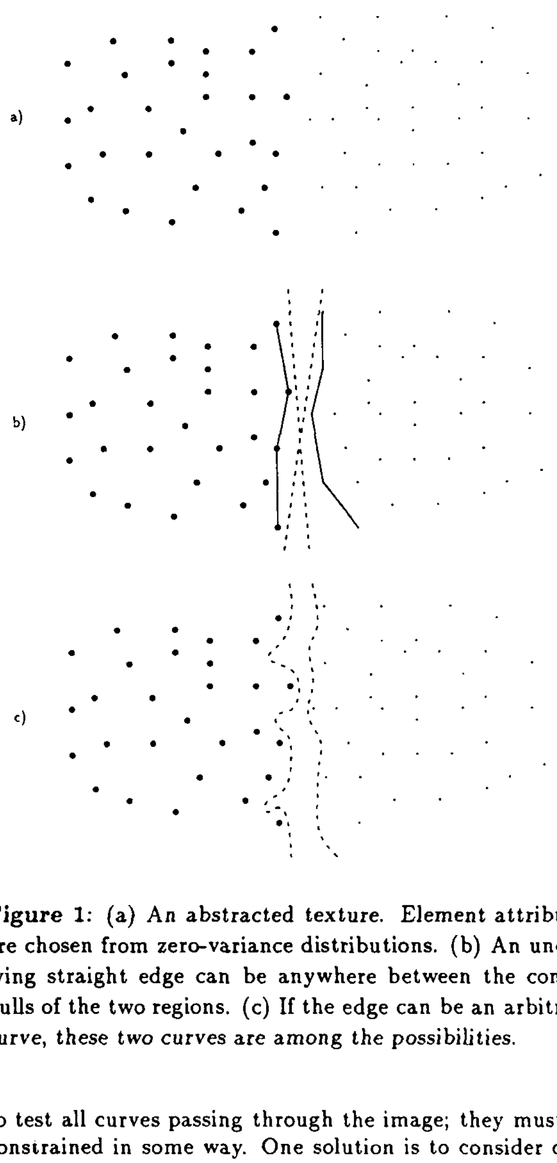


Figure 1: (a) An abstracted texture. Element attributes are chosen from zero-variance distributions. (b) An underlying straight edge can be anywhere between the convex hulls of the two regions. (c) If the edge can be an arbitrary curve, these two curves are among the possibilities.

2.1 Zero variance textures

Consider first the case in which the attribute variance in both LHS and RHS is zero, as in Figure 1a. In this situation, it is possible to classify elements based on their attribute value, and place the boundary between the clusters thus formed. However, it is rarely possible to localize the boundary exactly. In the simplest case, the boundary is known to be linear, and is then restricted to fall between the convex hull bounding the two regions, as shown in Figure 1b.

[1] Imagine two textures drawn on paper; cut a boundary in one texture and place it on the other. We wish to locate the boundary.

[2] This method of measuring the distance between texture regions is similar to the Mahalanobis distance [9], which is a variance-weighted measurement of difference between means. However, the Mahalanobis distance fails to consider the number of samples used to estimate the mean and variance; the statistical significance of any given difference in means grows with the sample size.

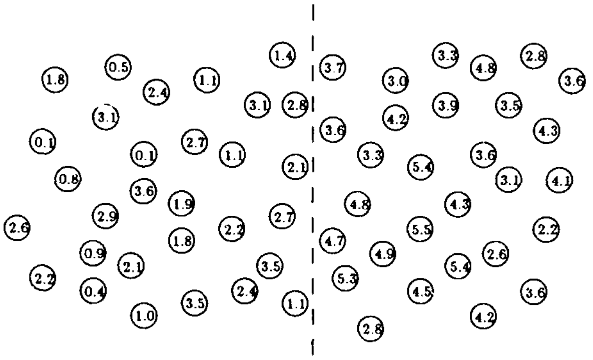


Figure 2: Elements from distributions with nonzero variance. Means on left and right are 2 and 4; both have variance 1.

When the boundary is not known to be linear, it is no longer restricted to fall between the convex hulls of the regions, but may be any of infinitely many curves separating the two textures; two such curves are shown in Figure 1c. Again, we generally wish to select the "simplest" of these curves, choosing a linear or piecewise linear boundary if possible; this choice depends on the measure of simplicity $S(C)$.

2.2 Nonzero variance textures

Next, consider the case where elements have nonzero variance, as shown in Figure 2. Now the elements cannot be classified with certainty, and it is no longer possible to restrict the boundary to the region between the clusters.

Suppose the edge is known to be linear and vertical, and we need only find its horizontal (x) position. Since the shape of the boundary curve C is constant, its simplicity $S(C)$ is constant, and the localization problem reduces to maximizing the texture difference $\Delta(C)$ across the curve. Denoting by C_x the curve at position x, the straightforward way to localize the boundary is to calculate the edge significance $\Delta(C_x)$ at several positions, with the best estimate for boundary location corresponding to the x that maximizes $\Delta(C_x)$. In the case of zero variance (Figure 1), $\Delta(C_x)$ is infinite when x is in the region between the two clusters; as x moves to the right of this region, the sample means grow closer, σ_L^2 increases rapidly, and $\Delta(C_x)$ decreases monotonically. When the variances are nonzero, there is still a peak in significance near the underlying edge, but due to random variation this peak may not coincide with the true edge, and the significance may not fall off monotonically on either side. Notice that $\Delta(C_x)$ is not continuous in x, but is instead a step function, since $\Delta(C_x)$ changes only when x crosses an image element. As a result, the x that maximizes $\Delta(C_x)$ can only be restricted to lie in an interval whose size is proportional to the average separation between elements. That is, localization accuracy increases with texture element density.

When both the orientation and position of the boundary are unknown, the best estimate of the texture boundary corresponds to the (x, θ) edge parameters that maximize the edge significance; one must search over the two-dimensional x and θ space. This is of course a much more complex problem than that of localizing in x alone. In general, there are an unknown number of image boundaries rather than just one. In this situation, there is no single "best estimate" for edge location, since when attributes have nonzero variance, $\Delta(C)$ may be significant for C falling between many nonoverlapping regions. The *scale problem* is one result of unknown boundary number and size. Figure 3 shows a simple example: at a large scale, the boundary at A is significant since it separates elements with small variance and widely different mean. On the other hand, the boundaries at B, surrounding the clusters of elements with zero variance, are highly significant if the regions over which distributions are measured are appropriately small. Depending on the application, it might be appropriate to detect the small- or large-scale boundaries, or both.

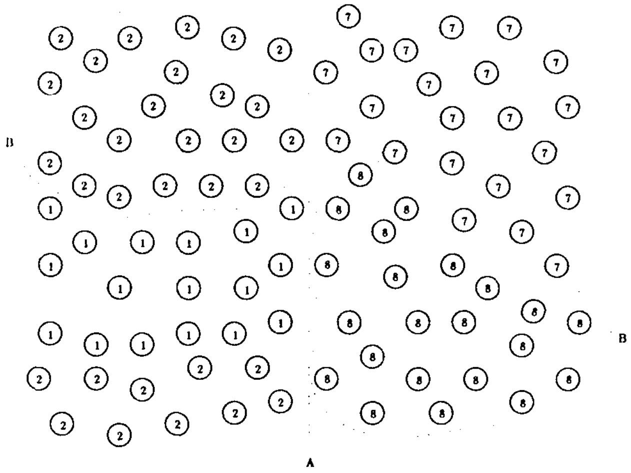


Figure 3: Illustration of one aspect of the scale problem. There is a large scale boundary at A, as well as small scale boundaries at B around the clusters of elements with zero variance.

2.3 Direct estimation of edge location

The problem of estimating the boundary position from the sampled attribute values is similar to estimating the mean of n samples u_i of a random variable U. From the u_i, one may directly compute a statistic, the sample mean, that estimates the mean; one may also compute the variance of this statistic, and derive a confidence interval for the mean. In texture edge localization, we wish to estimate the position of the underlying texture boundary and an associated variance of the estimate (c.f. [3]). The iterative procedure described above for estimating the edge location x corresponds to estimating the true mean μ of n samples by calculating the significance of the hypothesis $\mu = c$ for various

c, and choosing the c that maximizes the significance. It would be desirable to determine directly an estimate for edge location and its variance.

Unfortunately, there seem to exist no methods for this direct estimation. Consider the extremely simple one-dimensional case of Figure 2, that is, a set of random variables $(X_1, \ldots, X_T)$, where X_i follows distribution F for $i = 1, \ldots t$ and distribution G for $i = t+1, \ldots T$. The problem of determining t, the point at which the distribution changes, is known in statistics as the *change-point* problem, and has been extensively studied (e.g., [5,6,12,2]). The results relevant to our purposes may be summarized as: no explicit form exists for calculating the estimated change point $\hat{t}$. Instead, $\hat{t}$ must be computed sequentially, by testing the significance of the difference between estimated distributions on either side of conditional change-points t; the estimated $\hat{t}$ is that value of t which maximizes the significance. In addition, estimating the distribution of $\hat{t}$ is a nearly intractable problem, even asymptotically [12]. For normally distributed X_i with means μ_1 and μ_2, and common variance σ^2, Hinkley [5] shows that the variance of $\hat{t}$ depends asymptotically on $\Delta = |\mu_1 - \mu_2|/2\sigma$. When the difference in means $\mu_1 - \mu_2$ is large, $P(\hat{t} = t) \approx [\Phi(\Delta)]^2$, where Φ denotes the standard normal integral; thus the probability of getting the "right answer" increases with the difference between means, as one would expect.

3 A Localization Algorithm

The remainder of this paper discusses one simple approach to edge localization. This method should be regarded as an ad hoc approach whose intent is simply to demonstrate that the texture measures described in [11] suffice to segment real textured images in a reasonable manner.

The approach taken here involves calculating Δ for a fixed set of edge operators densely placed in x, y, θ space, and interpreting the pattern of responses. Our goal is to locate short linear edge segments, or 'edgels' (cf. [7]) that approximate the image edges. We choose an edgel length L, and assume that image edges have length at least L; shorter edges may be ignored. The input to the localization algorithm is a set of edge significance arrays $E_\theta(x, y)$, which contain at each point (x, y) the significance of a texture edge centered at (x, y) with orientation θ and length L. Each edge significance array E_θ contains information about the edges in directions near θ. The edge operator used here is described in [10] and depicted in Figure 4; the texture measures are described in [11]. For convenience, the width W of the edge operators is taken to be $L/2$, so the operator is square.

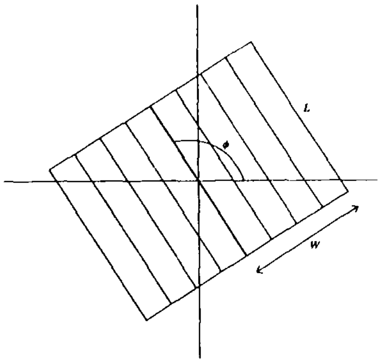


Figure 4: An edge operator. It divides its receptive field into N slices (here, $N = 4$) on either side of a hypothesized edge, as shown.

A texture edge in the image produces a ridge-shaped pattern in the edge significance array; the algorithm interprets these characteristic patterns as edges. The ridge shape results from the fact that the response of an edge operator generally is maximum when its parameters (x, y, θ) correspond with those of the underlying edge, and drops off as its position or orientation deviate from that of the edge. An edge operator is sensitive to edges when the orientations of the edge and the edge operator are sufficiently close (within a few degrees); the orientation of the resulting ridge follows the orientation of the underlying edge. Figure 5a shows an image with a horizontal edge, at 0 deg, and an oblique edge at -30 deg. An intensity edge operator with orientation $\theta = -20$ was applied to this image; Figure 5b shows the significance of the edge at positions near the corner. Notice that the edge operator response forms a ridge whose orientation is -30 deg, corresponding to the oblique edge. The operator responds weakly to the horizontal edge, since their orientations are quite different. Observe that the significance remains high for positions aligned with the edge but beyond its termination. Because the edge detector sums slices longitudinally, it is insensitive to terminations or corners of edges. We shall briefly discuss a method for dealing with corners below.

We detect ridges by searching for them in each of the edge significance arrays. Such ridges generally correspond to image edges. The union of these ridges over all orientations provides a first approximation to the estimated image edges. Figure 6 depicts the operation of this phase of operation of the system. Further processing is necessary to remove the portions of edges that extend beyond corners.

3.1 Ridge detection

To detect ridges, we fit a suitable function to small windows of E_θ. The significance of an edge typically drops off rapidly with distance from the underlying edge, falling to near zero, then rising to a low noise level. The minimum point generally occurs at about half the width of the edge operator. Figure 7 shows an example from a synthetic image with an intensity edge similar to that in Figure 5.

We use a quadratic function to describe the ridge shape. For a ridge centered at (x_0, y_0) with orientation ϕ, an appropriate functional form is

$$E_\theta(x, y) = ap^2(x, y) + bp(x, y) + c, \qquad (1)$$

where $p(x, y)$ is the perpendicular distance from the pixel at (x, y) to the line centered at (x_0, y_0) on the ridge (see

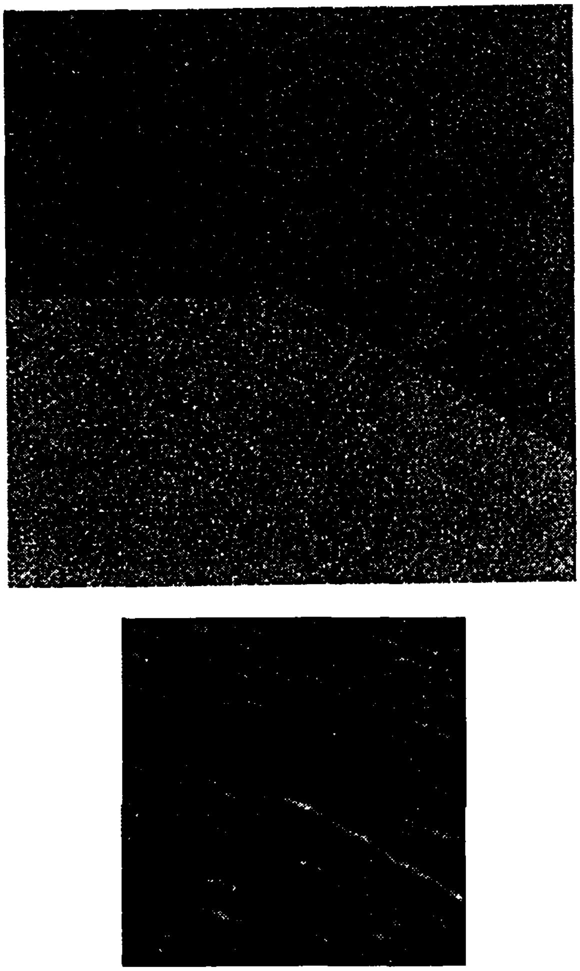


Figure 5: (a) An image containing an intensity edge with a 30-degree corner. Pixel intensity means are 0.3 on bottom, 0.6 on top; variance on either side is 0.15 (range 0 to 1). (b) Edge significances for an intensity edge operator of length 20, half-width 10, orientation −20 degrees. Bright pixels correspond to high significance.

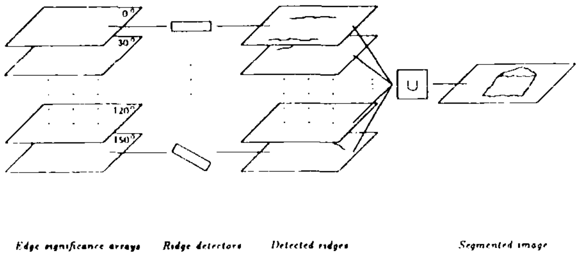


Figure 6: Illustration of localization phase of system.

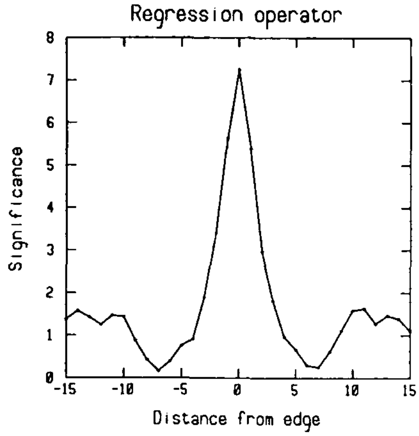


Figure 7: Intensity edge significance as a function of perpendicular distance from the underlying edge. Image parameters: $\mu_L = 0.3$, $\mu_R = 0.5$, $\sigma_L^2 = \sigma_R^2 = 0.15$. Operator parameters: $L = 40$, $W = 20$, $N = 5$. Averaged over 10 images.

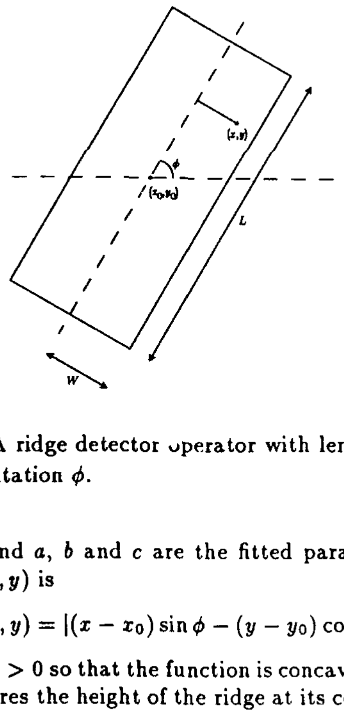


Figure 8: A ridge detector operator with length L, width W and orientation ϕ.

Figure 8), and a, b and c are the fitted parameters. The distance $p(x, y)$ is

$$p(x,y) = |(x - x_0)\sin\phi - (y - y_0)\cos\phi| . \qquad (2)$$

We require $a > 0$ so that the function is concave up. Parameter c measures the height of the ridge at its center, while a and b determine its steepness. Given a region of E_θ and a hypothesized position and orientation for the ridge, we wish to minimize the error of the data from the fitted function. Standard least-squares regression methods are used to find the coefficients a, b and c, and are not detailed here.

The ridge detection algorithm searches each array E_θ for ridges whose height c is over a threshold. The orientation of such ridges is found iteratively by searching for the orientation that results in the minimum error of fit. Since

E_θ contains information for edges near θ, the orientation search can be constrained to a few degrees in either direction from θ. The details of the ridge detection algorithm follow. Suppose the orientation difference between edge significance planes is α, typically about 30 deg.

1. (Find local maxima) Search through E_θ for local (8-neighbor) maxima $E_\theta(x, y)$ such that $E_\theta(x, y) > T_1$, where T_1 is a threshold. Repeat the following steps for each such maximum location (x_0, y_0).
2. (Find best ridge fit) Determine the ridge-function parameters a, b and c for ridges centered at (x_0, y_0), for several orientations ϕ ranging from $\theta - \alpha/2$ to $\theta + \alpha/2$. For each such ϕ, the fitted function considers points $E_\theta(u, v)$ that lie inside a rectangular window with half-width W, length L, and orientation ϕ, as shown in Figure 8; L is the same as the length of the edge operator used to create the edge significance arrays, and W is half the width of the edge operator. For each of these orientations determine the error of fit. Let ϕ_0 be the orientation at which the error is minimized, and c_0 be the height of the fitted ridge there.
3. (Ignore low ridges) If $c_0 < T_2$ (where T_2 is another threshold, the minimum ridge height) ignore this ridge as insignificant. Otherwise, the evidence suggests that there is an image edge centered at (x_0, y_0) with half-length L and orientation ϕ_0. Draw such a line in the image to mark it.

Figure 9 illustrates the working of this algorithm on the image of Figure 5. Simple intensity edge operators (with $L = 20$, $W = 10$, $N = 4$) were applied to the image at every location, at six orientations 10, 40, 70, 100, 130 and 160 degrees, forming the edge significance arrays E_θ. (Note that neither of the image edges is aligned with any of the edge detectors used; also, the program had no knowledge of edge position or orientation.) The edge significance arrays for $\theta = 10$ and $\theta = 130$ are shown in Figures 9a and 9b, with bright pixels corresponding to high significance; the edge significance array for $\theta = 160 = -20$ appears in Figure 5b. Notice that the ridges follow the image edges. Figure 9c shows the ridges found by the above algorithm, for all six orientations. This algorithm does a reasonable job of localizing the edges. It must be emphasized that any demonstration that purports to find "the edges" in an image, for comparison with human perception, necessarily involves thresholds on edge significance. The thresholds for the demonstrations in this paper were adjusted manually so that the detected edges correspond as nearly as possible with those perceived by humans.

3.2 Handling corners

Examination of Figure 9c shows that the detected ridges extend beyond the corner. The amount of overextension is typically about 1/3 the length of the ridge detector operator, but depends on the threshold T_2, with more overextension as T_2 decreases. When small operators are used, this overextension can often be ignored (as in [8]), but when larger operators are used, overextension becomes more of an issue, and it is desirable to suppress the ridges that extend beyond a tangent discontinuity in the edge.

The simple method used here relies on the fact that, at a corner, a ridge is intersected by a ridge of a different orientation, forming an "χ"-shaped pattern. In most cases, removing the two shortest subridges radiating from the intersection is sufficient to suppress the overextended ridges. Figure 9d, computed from Figure 9c, shows the results of such a computation; the corner is now more accurately localized.

3.3 Demonstration

The localization algorithm was applied to the textured image shown in Figure 10a, a foam square lying on an image of pigskin from Brodatz [4]. In the halftoned image reproduced in this paper it is difficult to see the shadows to the left and right of the foam square (due to multiple light sources) and the variations in texture within the foam itself. These aspects add to the difficulty of segmenting the image.

To summarize the segmentation process, the texture features described in [11] were computed over the upper portion of the image. The overall significance of texture edges was computed with edge operators of length $L = 25$, width $W = 12$; seven orientations (10, 36, 62, 88, 114, 140, 166 degrees) were used, providing the edge significance arrays E_θ. Figure 10b shows the results of the edge localization algorithm after corner suppression.

The orientation of the left and right edges of the square is about 78°; of the top, about 168°. The top edge is easily detected since there is a sharp texture boundary, no shadow, and the orientation of the edge and the edge operator at 166° differ by *only* 2°. The orientations of the left and right edges fall midway between the available edge detector orientations (62 and 88 degrees), making them more difficult to detect. In addition, the shadow edges are within one operator-width of the foam/background edge, which decreases the edge significance for the main edge. As a result, the left and right edges were not as well localized as the top edge. The texture edge operators are quite sensitive to texture differences, and detected some edges within the foam square and near its boundaries that are not easily visible in the halftoned image reproduced here. In spite of these difficulties, most of the dominant edges bounding the square were detected and localized.

4 Summary and Conclusions

This paper discussed some aspects of the problem of edge localization, and proposed a simple implementation of the ideas. We began by examining the localization problem for boundaries of very simple, abstract textures. We saw that even in this simple situation the localization problem is quite difficult. There is no direct way to estimate the boundary location even when its orientation is known; the location must be determined iteratively by finding the position that maximizes the significance of the difference between the textures on either side. There is generally no unique position that maximizes the significance; the edge

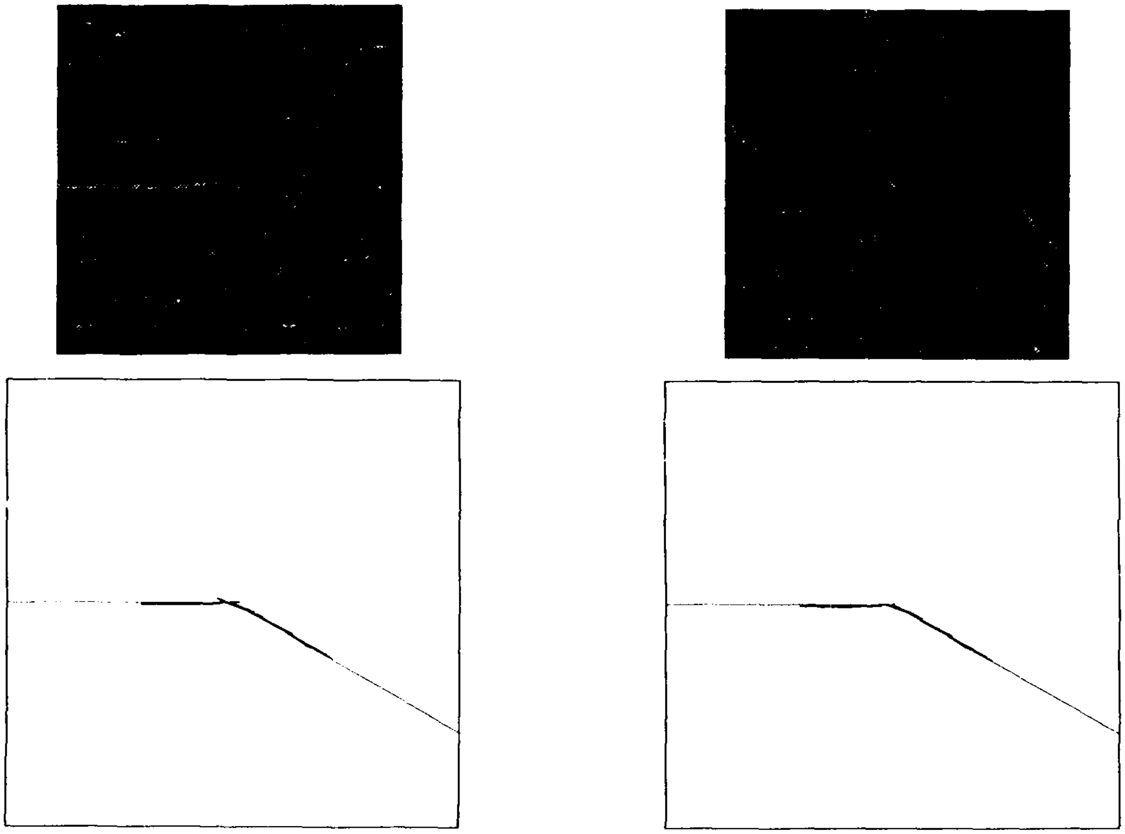


Figure 9: (a) Edge significanc array for edges in Figure 5, $\theta = 10$. (b) Edge significance array for $\theta = 130$. (c) Detected ridges; thresholds $T_1 = 2.5$, $T_2 = 4.1$. The image is removed for clarity; the thin lines represent the underlying edge. (d) Results after removing overextended portions of ridges.

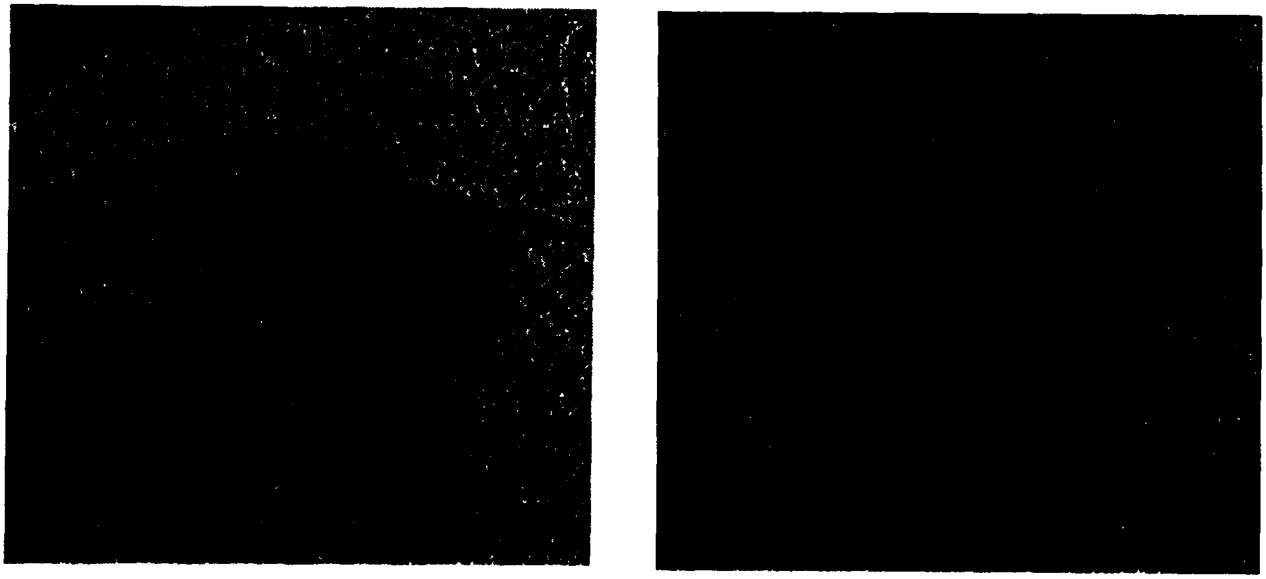


Figure 10: (a) The test image: a square piece of foam on an image of pigskin. No preprocessing was performed on this image. (b) Detected edges, with corner suppression. Thresholds $T_1 = 4.0$, $T_2 = 7.1$.

can only be localized to fall within a region whose size depends on the density of the texture elements. In general, the number of boundaries and their position and orientation are unknown, and we encounter difficult issues such as the scale problem. Localizing edges requires evaluating edge operators at many positions and orientations, and interpreting these edge significances to obtain a coherent description of the underlying edges.

A simple edge localization algorithm was proposed. It takes as input a number of edge significance planes E_θ which contain at each point (x, y) the significance of an edge centered at (x, y) in direction θ. Image edges are manifested as ridges in the edge significance arrays, with the orientation of the ridge matching the orientation of the image edge. The localization algorithm interprets these significance arrays, searching for patterns that correspond to image edges; only directions near θ need to be examined for ridges. These edge operators respond strongly when centered laterally on an edge, even when they extend beyond the termination of the edge. A simple method was discussed that removes many of these overextended edges.

The algorithm discussed here leaves many issues unresolved. The problem of measuring curve simplicity in a well-motivated way remains open. The scale problem is particularly difficult but has received little attention to date. We have not discussed linking of the detected edgels; for a discussion of that subject see [1,8].

5 Acknowledgements

I am grateful to Tom Binford for many valuable discussions, and to Aram Sogomonian for referring me to the statistical literature on the change-point problem. This research was supported by a subcontract to Advanced Decision Systems, S1093 S-1 (Phase II), "Knowledge Based Vision Task B," from a contract to the Defense Advanced Research Projects Agency.

References

[1] D.H. Ballard and C.M. Brown. *Computer Vision.* Prentice-Hall, Englewood Cliffs, NJ, 1982.

[2] P.K. Bhattacharya. Maximum likelihood estimation of a change-point in the distribution of independent random variables: General multiparameter case. *Journal of Multivariate Analysis*, 23:183–208, 1987.

[3] T.O. Binford. Inferring surfaces from images. *Artificial Intelligence*, 17:205–244, 1981.

[4] P. Brodatz. *Textures: A photographic album for artists and designers.* Dover, New York, 1966.

[5] D.V. Hinkley. Inference about the change-point in a sequence of random variables. *Biometrika*, 57:1–17, 1970.

[6] D.V. Hinkley. Time-ordered classification. *Biometrika*, 59:509–523, 1972.

[7] V.S. Nalwa and T.O. Binford. On detecting edges. *IEEE Trans. Pattern Anal. Machine Intell.*, PAMI-8(6):699–714, 1986.

[8] V.S. Nalwa and E. Pauchon. Edgel aggregation and edge description. *Comp. Vision, Graphics, and Image Proc.*, 40:79–94, 1987.

[9] J.C. Simon. *Patterns and Operators.* McGraw-Hill, New York, 1986.

[10] R. Vistnes. Computer texture analysis and segmentation. Submitted, 1987. An abridged version appears in *Proceedings, IEEE Conf. Comp. Vision*, Miami, 1987.

[11] R. Vistnes. Texture models and image measures for segmentation. In *Image Understanding Workshop*, 1988 (these proceedings).

[12] S. Zacks. Survey of classical and Bayesian approaches to the change-point problem: Fixed sample and sequential procedures of testing and estimation. In M.H. Rizvi, J.S. Rustagi, and D. Siegmund, editors, *Recent Advances in Statistics*, pages 245–269, Academic Press, New York, 1983.

SURFACE ORIENTATION FROM PROJECTIVE FORESHORTENING OF ISOTROPIC TEXTURE AUTOCORRELATION

Lisa Gottesfeld Brown and Haim Shvaytser*

Department of Computer Science
Columbia University
New York, NY 10027

Abstract

A new method for determining local surface orientation from the autocorrelation function of statistically isotropic textures is introduced. It relies on the foreshortening that occurs in the image of an oriented surface, and the analogous foreshortening produced in the texture autocorrelation function. This method assumes textural isotropy, but does not require the texture to be composed of texels or assume texture regularities such as equal area texels or equal spacings between texels. This technique is applied to natural images of planar textured surfaces and found to give good results in many instances. The simplicity of the method and its use of information from all parts of the image are emphasized.

Introduction

An important task that arises in many computer vision systems is the reconstruction of three-dimensional depth information from two-dimensional images. Of the many potential depth cues discussed in the literature, texture might be expected to play a particularly central role in the processing of certain classes of images such as those of natural outdoor scenes. Indeed, a variety of shape-from-texture algorithms have been proposed[1].

The determination of surface orientation from textural cues has been based on two general techniques. Gradient methods, exemplified by the work of Gibson[2], rely on changes in texture properties such as the density of texels as the surface recedes from the observer. It is also possible, however, to deduce surface orientation from purely local properties of the observed texture. Our work follows this latter approach, and bears many similarities to the algorithm presented in the pioneering paper of Witkin[3].

Witkin begins with the assumption that the texture is isotropic, that is, that statistically speaking the texture has no inherent directionality. The distribution of edge directions obtained from such a texture will therefore be flat. If, however, the textured surface is viewed obliquely, projective foreshortening — a purely local phenomenon — distorts this distribution in a well-defined way. Witkin thus proposes that a histogram of edge directions constructed from the image in question be used to determine surface orientation via a maximum-likelihood fit.

The method proposed here is also based on the assumption of textural isotropy, but uses the projective distortion of the texture autocorrelation function as an orientation cue rather than the projective distortion of Witkin's edge-direction histogram. The autocorrelation of an oriented texture is foreshortened in a way identical to the foreshortening of the image itself, and the amount and direction of the foreshortening are conveniently measured by the moments of the autocorrelation function. Potential advantages of this approach are the elimination of the arbitrariness associated with the choice of edge-detection algorithm and the fact that it uses information from all parts of the texture rather than just the edges.

The next section contains a quantitative treatment of projective foreshortening of texture autocorrelation, and culminates in a formula expressing surface orientation as a function of the moments of the texture autocorrelation. The third section provides the practical details of an algorithm based on this approach, and describes the results it yielded when applied to textures found in common outdoor scenes.

Foreshortening of Texture Autocorrelation

Let us begin with the terminology to be used in this and the subsequent section. The orientation of a surface (with respect to the line of sight) will be given in terms of the slant and tilt parameters, (σ, τ), introduced by Witkin[3]. The slant of a surface, σ, is defined as the angle between the normal to the surface and the line of sight, That is, σ is the amount the surface slants away from being parallel to the image plane. We will take σ to run between 0° and 90°. The tilt, τ, specifies the direction in which the surface is slanted, and is defined as the angle between the x-axis of the image plane and the projection into the image plane of the normal to the surface. We will take τ to run between $-180°$ and $180°$. For example, a surface parallel to the image plane has zero slant and an undefined tilt.

Let an image be specified by a gray-scale function $F(\vec{r})$, where $\vec{r} = (r_1, r_2)$ denotes a point in the image plane. The image of a textured plane that is viewed head on, *i.e.*, that is perpendicular to the line of sight, will be denoted by $F_{\perp}(\vec{r})$. $F_{(\sigma,\tau)}(\vec{r})$ is then defined as the image produced by the same plane when it is given orientation (σ, τ) with respect to the line of sight.

The autocorrelation of an image, $A(\vec{r})$, is defined as

$$A(\vec{r}) = \int (F(\vec{r}\,') - \bar{F})\,(F(\vec{r}\,' + \vec{r}\,) - \bar{F})\,d\vec{r}\,', \tag{1}$$

where $\bar{F}$ is the mean of F. The autocorrelation is conventionally normalized such that $A(\vec{0}) = 1$, but our purposes do not

*This work was supported in part by DARPA grant # N00039-84-C-0165. One of us (H.S.) held a Weizmann fellowship during the course of this work.

require a specific choice of normalization to be made. The head-on autocorrelation, $A_\perp$, and the oriented autocorrelation, $A_{(\sigma,\tau)}$, correspond to the images $F_\perp$ and $F_{(\sigma,\tau)}$, respectively.

Finally, we introduce the autocorrelation moment matrix[1],

$$\mu = \begin{pmatrix} \mu_{11} & \mu_{12} \\ \mu_{21} & \mu_{22} \end{pmatrix},$$

defined as

$$\mu_{ij} \equiv \int r_i r_j A(\vec{r})\, d\vec{r}. \qquad (2)$$

The moment matrix is trivially symmetric; $\mu_{12} = \mu_{21}$. Since the region of integration in equation (2) is notionally infinite (although it corresponds in practice to a finite sum over pixels) we require $A(\vec{r})$ to fall off sufficiently rapidly at large distance that the moments be well-defined. The randomness found in natural textures suffices to produce the desired behavior. A highly regular pattern, however, can have an autocorrelation that remains large at large distance; the algorithms presented here cannot be applied directly in such a situation. The autocorrelation moment matrices derived from the image of a textured plane viewed head on and the same plane with orientation (σ, τ) will be denoted $\mu_\perp$ and $\mu_{(\sigma,\tau)}$, respectively.

We now discuss how projective distortion affects the autocorrelation moment matrix of an image, and how this information can be used to determine the orientation of a textured plane. Let $F_\perp(\vec{r})$ be the head-on image of a textured plane. Now consider $F_{(\sigma,\tau)}(\vec{r})$, the image produced by the same planar surface when viewed obliquely with slant and tilt parameters (σ, τ). We make the simplifying assumption that the distance between the plane and observer is very large in comparison with the linear size of the portion of the plane under consideration. The image of the oriented plane is therefore given by orthographic projection:

$$F_{(\sigma,\tau)}(\vec{r}) = F_\perp(\mathbf{M}^{-1}(\sigma,\tau)\vec{r}), \qquad (3)$$

where

$$\mathbf{M}(\sigma,\tau) \equiv \begin{pmatrix} 1 + (\cos\sigma - 1)\cos^2\tau & (\cos\sigma - 1)\cos\tau\sin\tau \\ (\cos\sigma - 1)\cos\tau\sin\tau & 1 + (\cos\sigma - 1)\sin^2\tau \end{pmatrix}. \qquad (4)$$

Notice that

$$\mathbf{M}(\sigma,\tau) = \mathbf{R}_\tau \mathbf{M}(\sigma,0)\mathbf{R}_{-\tau}, \qquad (5a)$$

with

$$\mathbf{R}_\tau = \begin{pmatrix} \cos\tau & -\sin\tau \\ \sin\tau & \cos\tau \end{pmatrix}, \qquad (5b)$$

and

$$\mathbf{M}(\sigma,0) = \begin{pmatrix} \cos\sigma & 0 \\ 0 & 1 \end{pmatrix}. \qquad (5c)$$

That is, $\mathbf{M}(\sigma,\tau)$ is the matrix which foreshortens the vector $\vec{r}$ by a factor of $\cos\sigma$ along the direction of tilt, τ, while leaving the direction perpendicular to τ unchanged. We should emphasize that $\mathbf{M}(\sigma, \tau + 180°)$ (or equivalently $\mathbf{M}(-\sigma,\tau)$) and $\mathbf{M}(\sigma,\tau)$ are equal. It is clear, therefore, that our method will yield surface orientation only up to an overall $\tau \leftrightarrow \tau + 180°$ ambiguity.

The image autocorrelation transforms identically to the image under orthographic projection:

$$\mathbf{A}_{(\sigma,\tau)}(\vec{r}) = \mathbf{A}_\perp(\mathbf{M}^{-1}(\sigma,\tau)\vec{r}). \qquad (6)$$

To relate $\mu_{(\sigma,\tau)}$ to $\mu_\perp$ we evaluate the integral in equation (2) via the change of variables $\vec{r} \to \vec{r}\,' = \mathbf{M}(\sigma,\tau)\vec{r}$, and (using the result $\det\mathbf{M}(\sigma,\tau) = \cos\sigma$) obtain

$$\mu_{(\sigma,\tau)} = \cos\sigma\mathbf{M}(\sigma,\tau)\mu_\perp\mathbf{M}(\sigma,\tau).$$

In order to extract the slant and tilt parameters from μ some *a priori* information about the texture is required. In this work we will assume that, in a statistical sense, the texture is isotropic,[2] and hence that the moment matrix is a multiple of the identity, $\mu_\perp = \xi\mathbf{1}$. Given this assumption, we now have

$$\mu_{(\sigma,\tau)} = c\mathbf{M}^2(\sigma,\tau), \qquad (7)$$

where the constant is given by $c = \xi\cos\sigma$.

Equation (7), which relates the autocorrelation moment matrix of the image of a (directionally homogeneous) textured surface to the slant and tilt parameters that specify its orientation, forms the basis of our shape-from-autocorrelation algorithm. We complete our discussion of the underpinnings of this algorithm by giving the slant and tilt parameters as explicit functions of the matrix $\mu_{(\sigma,\tau)}$. From equation (5c) $\cos^2\sigma$ is given by the ratio of the smaller to the larger eigenvalue of $\mu(\sigma,\tau) \propto \mathbf{M}^2(\sigma,\tau)$:

$$\sigma = \arccos\sqrt{\frac{\mu_{11} + \mu_{22} - \sqrt{(\mu_{11} - \mu_{22})^2 + 4\mu_{12}}}{\mu_{11} + \mu_{22} + \sqrt{(\mu_{11} - \mu_{22})^2 + 4\mu_{12}}}}. \qquad (8a)$$

The expression for the tilt parameter, τ. follows from equation (4):

$$\tau = \tfrac{1}{2}\arctan\frac{2\mu_{12}}{\mu_{11} - \mu_{22}}. \qquad (8b)$$

To summarize: equation (8), our principal result, gives the orientation of a textured surface as a function of its autocorrelation moment matrix, provided

- textural isotropy is assumed,
- the autocorrelation falls off sufficiently rapidly with distance,
- orthographic projection is assumed, and
- additional information is available to resolve the $\tau \leftrightarrow \tau + 180°$ ambiguity.

Implementation and Results

To test this method, images were made of textures that seemed reasonably isotropic. In each case a single planar surface was photographed from a sufficient distance that orthographic projection was a good approximation, and that the entire image consisted of an image with a single orientation. We used textures commonly found in natural outdoor scenes, such as grass, rocks, dirt and leaves. Because of their practical importance in navigation, images of roads, sidewalks, and pebbled pathways were also included.

[1] The use of moments in vision processing has been discussed by Hu in reference [4]. We note that our notations differ   the correspondence is given by $(\mu_{11}, \mu_{12} = \mu_{21}, \mu_{22}) \leftrightarrow (m_{20}, m_{11}, m_{02})$.

[2] It should be clear that this assumption need not be strictly satisfied. It suffices to make the weaker assumption that $\mu_\perp = \xi\mathbf{1}$, that is, that any directional inhomogeneities the texture may have are not of the sort that mimic an oblique viewing angle, and hence do not show up in the moment matrix.

The following simple scheme permitted the actual orientation to be conveniently determined. Two photographs of each surface were taken, identical except that in one of the pictures a flat circular object was placed on the surface. The picture without the object was used as input to our shape-from-autocorrelation algorithm, while measurements of the foreshortened image of the circular object in the second picture gave the true surface orientation.

The photographs were digitized to yield 256 × 256 8-bit gray-scale images, and the autocorrelation was computed as the Fourier transform of the power spectrum of the image. Since the Fourier transform can be computed with $O(n \log n)$ cost, where n is the size of the number of pixels in the image, and the moments computed with $O(n)$ cost, the entire procedure has a complexity of $O(n \log n)$. Thus, for a small cost above the optimal, the method uses all the information in the image.

When computing the moments the integration in equation (2) is replaced by a summation, with the autocorrelation nominally being summed over all possible separations. To the extent that the autocorrelation falls off rapidly, the result should be insensitive to the precise region of summation. However, realistic images do not obey this assumption in a robust way, and statistical noise in the autocorrelation at large separations can cause excessive error in the values of the moments. To alleviate this problem we sum only over autocorrelations which exceed some threshold value, effectively restricting the moment sum to relatively small separations. The threshold value is chosen to be the value of the autocorrelation averaged over a ring with an empirically determined radius. In our experiments, a radius of 10 pixels was found to give good results. Figure 1 shows how this radius effects the slant and tilt estimates for four typical images (D,H,I and K). For each image the 'O' marks the radius value in which the computed (σ, τ) matches the actual value. As seen from these figures the threshold radius must be chosen intelligently, but the choice is not particularly delicate — any threshold radius in the range ~ 5 *to* 25 would yield similar results.

A summary of the results is shown in Figure 2. In this plot each orientation is represented by a point in polar coordinates where the slant is the distance from the center and the tilt is the angle. For each image (A-K) the error between the actual measurement of the surface orientation and the value predicted by the method is shown. Considering the difficulty in discerning orientations of homogeneously textured surfaces without perspective or world-knowledge cues, all of the results except images F and D were very reasonable.

The four representative images illustrate the technique and its behavior in more detail in Figure 3. The large picture in the upper left hand corner of each of the four examples shows a 512 × 512 color image of the textured surface with the flat circular object laying upon it. As was just discussed, from the foreshortening of the circular object in this image, the actual surface orientation was determined. The picture in the upper right hand corner shows the 256 × 256 black and white subimage of the larger image without the circular object. It is the autocorrelation of this image that was used to compute an estimate of the surface orientation. The actual and computed orientations are given at the bottom of each figure. The picture in the lower right corner shows the central 64 × 61 region of the autocorrelation. The predicted foreshortening of the autocorrelation is clearly seen, and the elliptic shape of the foreshortened autocorrelation corresponds closely to the foreshortened image of the circular object placed on the surface. It should be observed, that from the pictures used for the computation, it is virtually impossible for most people to perceive the surface orientation while nevertheless the method is often successful. Even in the road example where the predicted orientation is poor, the autocorrelation shows the expected properties.

Concluding Remarks

We have developed a new shape-from-texture algorithm that uses the effects of foreshortening on the autocorrelation of an image. The method assumes textural isotropy, that is, that the statistical properties of the texture have no inherent directionality, and that the texture is orthographically projected onto the image plane. The performance of this algorithm is quite good; the empirical tests presented in the previous section show that this algorithm gives good estimates of surface orientation when applied to a variety of natural textures. This method is simple and well-defined, and avoids, for example, assumptions concerning texels and the arbitrariness associated with edge-detection. A particular advantage of this approach is its use of information from all parts of the image; all pixels contribute to the final orientation estimate.

Future extensions of this method might include weakening the assumption of textural isotropy through the use of multiple views, techniques involving gradients of the texture autocorrelation, or use of a knowledge system to provide *a priori* information about the head-on texture autocorrelation. Surface curvature should be measurable by applying the method patch-wise over an image. A parallel implementation using image shifting to compute the relevant portion of the autocorrelation should lead to substantial increases in the speed of the algorithm.

Acknowledgments

We would like to thank John Kender for his support and encouragement. We would also like to thank Monnett Hanvey for the use of her photographic equipment and expertise.

References

[1] J. Aloimonos and M.J. Swain "Shape from Texture," *Proc. of the Tenth International Joint Conf. on A.I.*, IJCAI, (1985).

R. Bajscy and L.Lieberman "Texture Gradient as a Depth Cue," *Computer Graphics and Image Processing* **5** (1976), 52-67.

K. Ikeuchi "Shape from Regular Patterns (an Example from Constraint Propagation in Vision)," *Proc. of the International Conf. on Pattern Recognition* (1980), 1032-1039.

J.R. Kender *Shape from Texture*, PhD thesis, CMU 1980.

Y.Ohta, K.Maenobu and T. Sakai "Obtain Surface Orientation from Texels under Perspective Projection," *Proc. of the Seventh International Joint Conf. on A.I.*, IJCAI, (1981).

[2] J. J. Gibson *The Perception of the Visual World*. Cambridge, MA: Riverside Press, 1950.

[3] A. P. Witkin, "Recovering Surface Shape and Orientation from Texture," *Artificial Intelligence* **17** (1981), 17-45.

[4] M. K. Hu, "Visual Pattern Recognition by Moment Invariants," *IRE Trans. Inform. Theory* **IT-8** (Feb. 1962), 179-187.

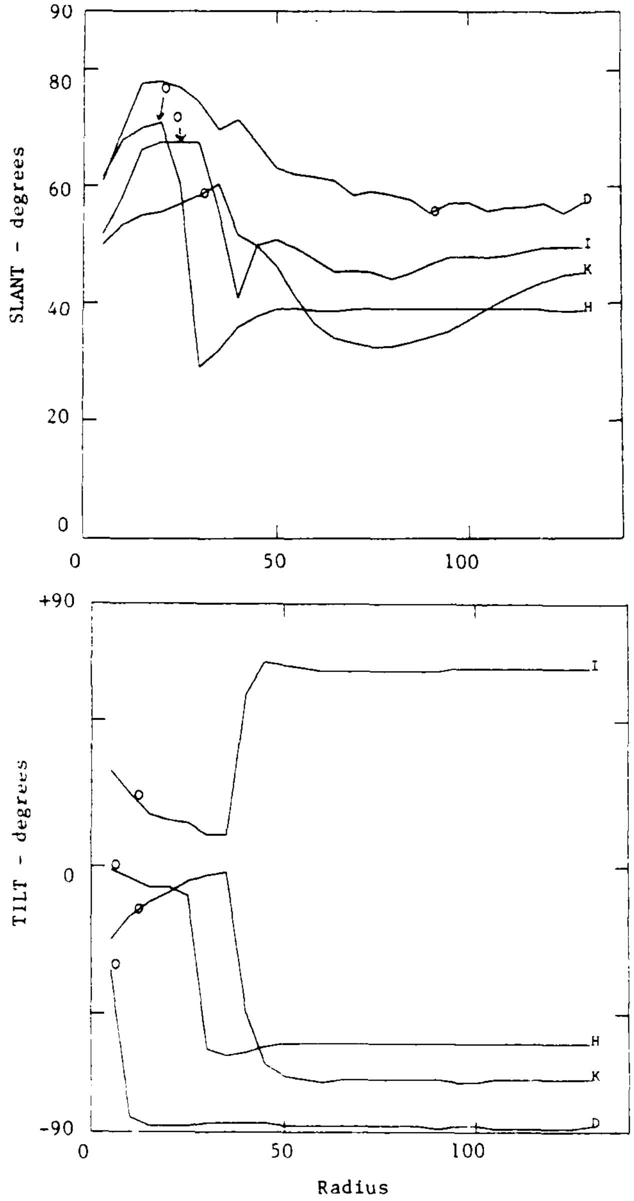


Figure 1. Effects of Thresholding Radius on the Elimination of Autocorrelation Noise for Four Typical Images. The small circle marks the radius value in which the computed slant or tilt matches the actual value.

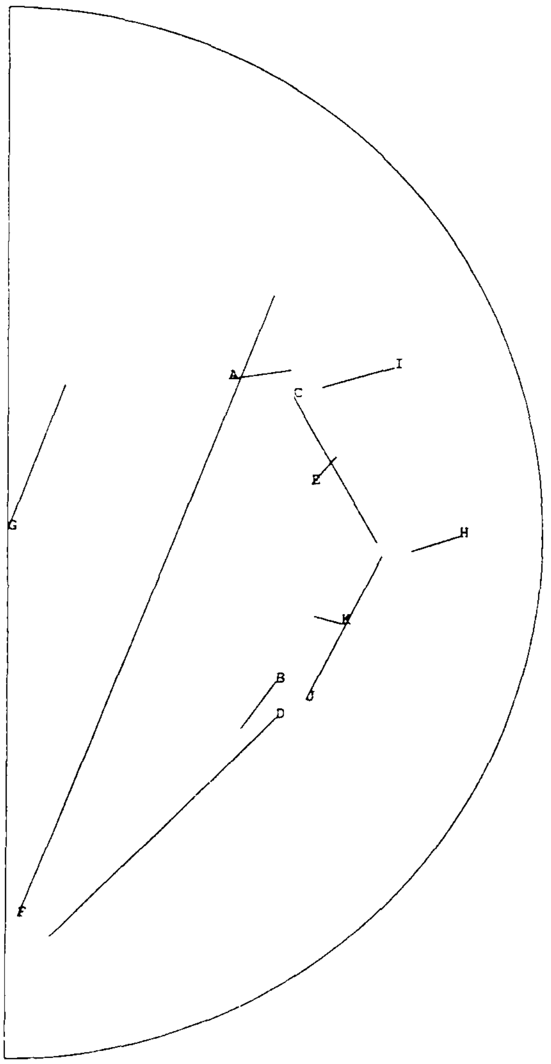
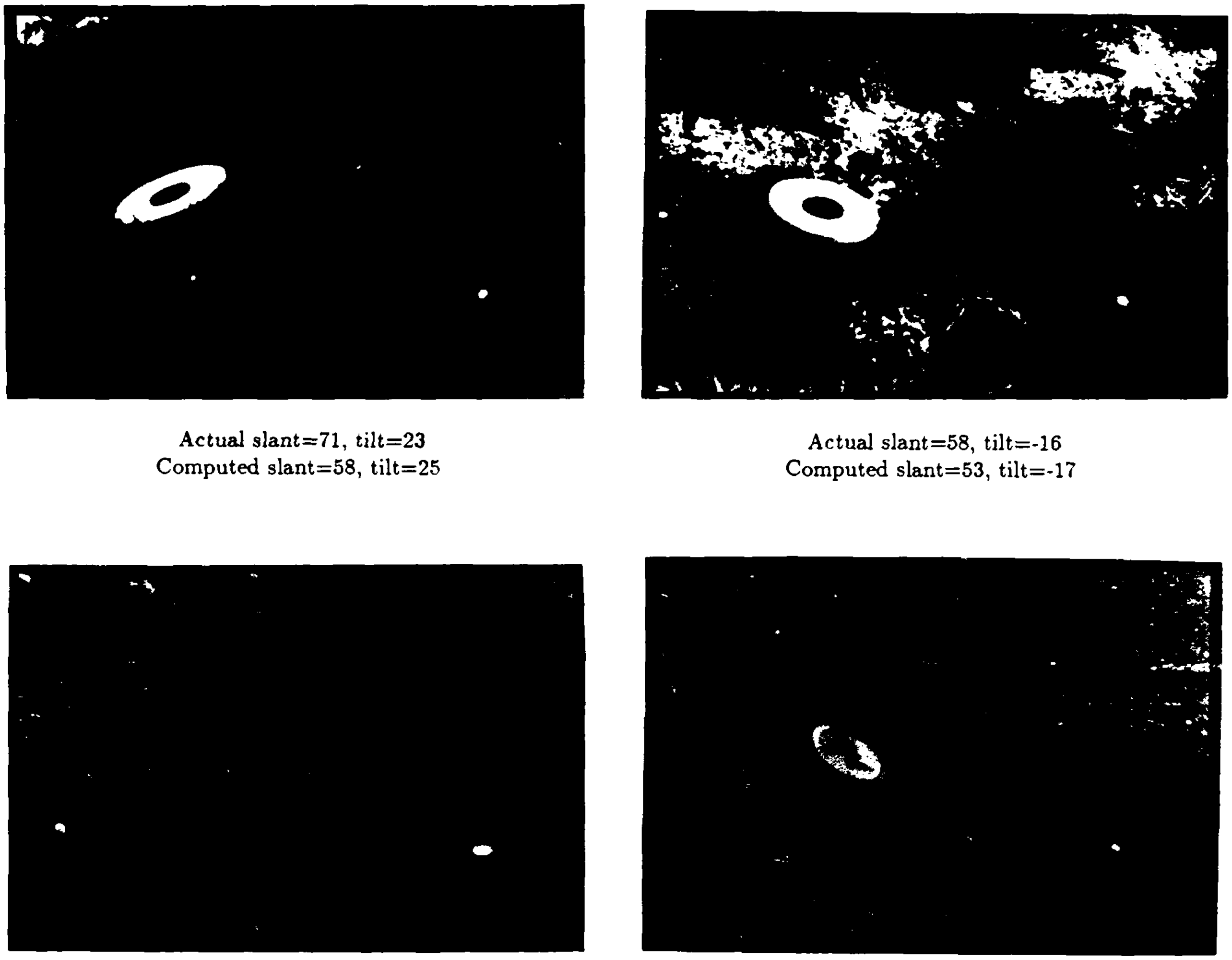


Figure 2. Polar Representation of Error Between Actual and Computed Surface Orientation for Eleven Pictures. The slant is the distance from the center and the tilt is the angle.



Actual slant=71, tilt=23
Computed slant=58, tilt=25

Actual slant=58, tilt=-16
Computed slant=53, tilt=-17

Actual slant=76, tilt=-1
Computed slant=68, tilt=-4

Actual slant=55, tilt=-35
Computed slant=70, tilt=-85

Figure 3. Results of Autocorrelation Method. The large picture in the left hand corner of each of the four figures shows a 512 × 512 color image of the textured surface with the flat circular object laying upon it from which the actual surface orientation was determined. The picture in the upper right hand corner shows the 256 × 256 black and white subimage of the larger image without the circular object which was used to compute an estimate of the surface orientation. The picture in the lower right corner shows the central 64 × 64 region of the autocorrelation.

Labeling Polyhedral Images

Van-Duc Nguyen

General Electric Corporate Research and Development Center
Schenectady, NY 12301

Abstract

Recognizing that a polyhedral image is a segmentation of the plane into connected regions, called faces, joined by vertices and edges, the input image is represented as a planar network of nodes (vertices, edges, faces) linked by adjacency links. By introducing junction-pairs and junction-loops as labelings of edges and faces, globally consistent labelings of the whole image can be found through achieving node and arc consistencies at all nodes and links in the network. This proves that labeling of polyhedral images and line drawings is linear in the number of edges in the image, and exponential only in the maximum number of edges intersecting at a node. Test examples on ideal and real polyhedral images are illustrated.

1. Image Segmentation

The hidden line view of a finite 3D polyhedral scene is a finite 2D image. If the hidden line view is completely inside the image boundary, then the closed object faces project into zero or many closed image faces. The other faces of the image must come from the background of the 3D scene. Only the face surrounding the hidden line view is open and needs to be bounded by the image boundary. All the other faces are closed by the edges in their boundaries. If the image frame overlaps the hidden line view, then some edges and faces of the image are cut off and bounded by the image boundary. The segments and end points that are on this boundary cannot be considered as valid junctions because the faces around them are not completely defined. So, we must exclude the image boundary from labeling and segmentation.

Faces, edges, and vertices are connected sets in R^2. If we exclude the image boundary, then we can define the interior component of a set S as the set of points of S not on the boundary ∂S and not on the image boundary ∂I. For example, the interior of a vertex inside ∂I is a point. The interior of an edge (resp. face) is an open and connected segment (resp. region). We note that no point of the image is in the interior of two distinct topological sets, or formally:

Theorem 1 *The disjoint union of the interior components of all vertices, edges, and faces of the image is equal to the open set inside the image boundary.*

The image is segmented into subsets of R^2, which have adjacent but non-overlaping interior components. This segmentation can be represented by a planar network of linked nodes. The nodes are the vertices, edges and faces, or topological subsets of the image. The links describe the adjacency relationship between these topological sets.

2. Node Consistency

Consider the following scenario. We cut n distinct pictures with one single template to create n puzzles and pile them on top of each other. The template cuts a picture into m pieces, so there are m piles. Each pile has n pieces, all with the same shape but with different picture segments on them. Suppose *we shuffle the n pieces within each pile, then we remove at* random a few pieces from each pile. You are now asked to find all complete pictures from the m piles. Waltz (1975) showed that the most efficient way to tackle this task is to compare pieces from one pile against pieces from its adjacent pile, and prune away all pieces that can not possibly match. You stop when there is one pile that is empty, in which case there is no complete picture, or when there is no further pruning, in which case you have found all the complete pictures. Each complete picture can be recovered by picking the first piece from any starting pile, then find the adjacent pieces that match the first piece from each pile adjacent to the first pile, and so on, until we cover all the pieces in the template.

Taking the hint from the above scenario, our task is to define local labelings at each node of the image, and prove that global consistency can be achieved from node and arc consistencies at all nodes and links of the image. Huffman (1971) and Clowes (1971) first introduced junctions as local labelings at vertices. Representing as a pair or triple of edge-labels, a junction completely describes the local volume at a trihedral vertex (L, Fork, Arrow) or an occlusion (T). The local labeling at an edge E must completely describes the local volume around edge E up to its two end vertices and its two adjacent faces. The obvious choice is a set of edge-labels for edge E and all the edges intersecting E. To facilitate arc consistency between an edge and its end vertices, we choose to represent the local labeling at an edge by a junction-pair. Similarly, the local labeling at a face F is a junction-loop, which contains edge-labels for all the boundary edges in ∂F and all the edges intersecting the face boundary ∂F.

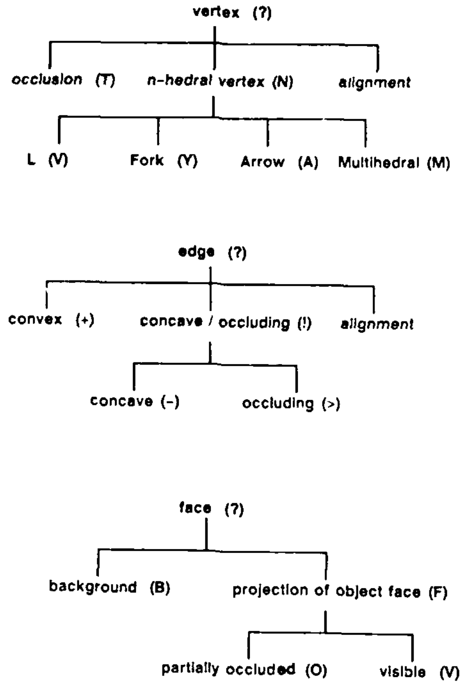


Figure 1: Label hierarchies for vertices. edges, and faces.

Figure 1 show the label hierarchies for the vertices, edges, and faces. Figure 2 show examples of junction, junction-pair, and junction-loop which completely describe the local volume at a vertex, edge, and face. The junctions for occlusion and trihedral vertices (Huffman 1971, Clowes 1971) are precomputed and stored in a dictionary. Junctions and vertex-labels are related by a $\subseteq$ relation, which describes node consistency at vertices. The junctions for multihedral vertices with zero or one hidden face can be enumerated depth-first from the Fundamental Junction Equation relating the orientation of the edges, the ordering of the faces, and their respective gradient points around a vertex (Malik 1985).

The junction-loops of a face F are found by a depth-first search for all consistent labelings of the vertices around the face boundary δF. Consistency means that the junctions of any two consecutive vertices must have the same edge-label for the edge connecting these two vertices. Each junction-loop found from depth-first search must be checked to see if the edge-labels of the boundary edges are all consistent with some face-label of F. This later is node-consistency enforced by the $\subseteq$ relation, relating junction-loops and face-labels. The enumeration of the junction-pairs also requires a depth-first search for all pairs of junctions that have the same edge-label for the common edge.

It is well known that there are no dangling edges and vertices in a projection of a polyhedral scene. Each edge has two distinct faces, and each vertex has as many faces as edges joined to it. So the number of vertex-edge, edge-face, and face-vertex links is *each at most twice the number of edges, and the number of nodes in the image* is at most three times the number of edges. With a complete search at each node to enumerate all the local labelings that are node-consistent, we deduce that:

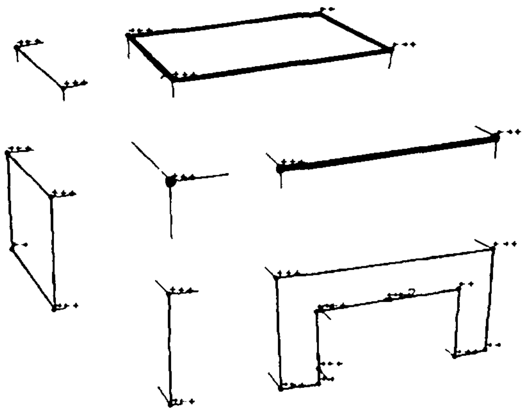


Figure 2: Junctions, junction-pairs, and junction-loops are respectively local labelings around a vertex, an edge, and a face.

Complexity 1 *The enumeration of all local labelings at all nodes has time complexity linear in the number of edges in the image and exponential in the maximum number of edges intersecting at a node.*

3. Arc Consistency

Constraints at all nodes are captured in the enumeration of all local labelings that satisfy node labels. These constraints are propagated over all links by pruning adjacent local labelings that can not possibly match. Let's describe the constraint satisfaction and propagation (CSP) over vertex-edge-face links, after all the local labelings are enumerated.

Algorithm 1 (Parallel CSP over vertex-edge-face links)

```
Initialize Vertex-Fifo, Edge-Fifo, and Face-Fifo respectively with all
vertices, edges, and faces in the image.
Loop until (and (emptyp Vertex-Fifo) (emptyp Edge-Fifo) (emptyp
Face-Fifo)) do

  1. Loop until (emptyp Vertex-Fifo) do
     Let $V$ := (pop Vertex-Fifo).

       a. Loop for all edge $E_i$ in Edges(V) do
          Restrict junction-pairs of $E_i$ with junctions of $V$.
          If the set of junction-pairs of edge $E_i$ has been restricted
          then (insert $E_i$ Edge-Fifo).

       b. Loop for all face $F_i$ in Faces(V) do
          Restrict junction-loops of $F_i$ with junctions of $V$.
          If the set of junction-loops of face $F_i$ has been restricted
          then (insert $F_i$ Face-Fifo)

  2. Loop until (emptyp Edge-Fifo) do
     Let $E$ := (pop Edge-Fifo).

       a. Loop for all face $F_i$ in Faces(E) do
          Restrict junction-loops of $F_i$ with junction-pairs of $E$.
          If the set of junction-loops of face $F_i$ has been restricted
          then (insert $F_i$ Face-Fifo).

       b. Loop for all vertex $V_i$ in Vertices(E) do
          Restrict junctions of $V_i$ with junction-pairs of $E$.
          If the set of junctions of vertex $V_i$ has been restricted
          then (insert $V_i$ Vertex-Fifo).
```

```
3. Loop until (emptyp Face-Fifo) do
   Let $F$ := (pop Face-Fifo).
      a. Loop for all vertex $V_i$ in Vertices(F) do
         Restrict junctions of $V_i$ with junction-loops of $F$.
         If the set of junctions of vertex $V_i$ has been restricted
         then (insert $V_i$ Vertex-Fifo).
      b. Loop for all edge $E_i$ in Edges(F) do
         Restrict junction-pairs of $E_i$ with junction-loops of $F$.
         If the set of junction-pairs of edge $E_i$ has been restricted
         then (insert $E_i$ Edge-Fifo).
```

Algorithm 2 (Restrict junctions of V with junction-pairs of E)

```
Loop for all $J$ in Junctions(V) do
   if Loop for all $P$ in Junction-Pairs(E)
         never $J \subseteq aref(P,V)$
      then if Loop for all $P$ in Junction-Pairs(E)
               thereis $aref(P,V) \subseteq J$
         then replace $J$ by its sub-junctions
         else delete $J$ from Junctions(V).
```

Complexity 2 *Let m_J, m_P, m_L be the maximum number of junctions, junction-pairs, junction-loops at any vertex, edge, face in the image. Without the hierarchy of edge-labels, the time complexity of constraint satisfaction and propagation over vertex-edge-face links of a polyhedral image I is:*

$$O\left(m_J\, m_P\, m_L \left(\frac{m_P + m_L}{m_J} + \frac{m_L + m_J}{m_P} + \frac{m_J + m_P}{m_L}\right) |Edges(I)|\right)$$

If there is no hierarchy of edge-labels, the consistency relation is = instead of ⊆, and there is no label refinement. The set of labelings have the worst case cardinality, and Algorithm 2 has time complexity $O(m_J m_P)$ time. We note that the number of edge-vertex arcs is twice the number of edges in the image. Each edge can be in Edge-Fifo m_P times, so in the worst case the constraint procedure in Algorithm 2 is called at most $2m_P|Edges(I)|$ times, and the edge-vertex arc consistency of Loop 2.b costs $O(m_P^2 m_J|Edges(I)|)$ time. Similarly, we prove that the other arc-consistencies have time complexity linear in the number of edges in the image, and polynomial in the number of local labelings. ■

4. Global Consistency

We have shown how the constraints are satisfied at each node and propagated to the neighbors using node and arc consistencies. A node is locally consistent if and only if it has local labelings that satisfy node consistency and all arc consistencies with neighboring nodes. Let's now prove that local consistency at all nodes implies global consistency, or formally:

Theorem 2 *After CSP, the polyhedral image I has globally consistent labelings if and only if all the vertices, edges, and faces of I have locally consistent labelings.*

We show by induction that we can enumerate all globally consistent diagrams of the image. The proof will highlight the necessity of local constraints captured by junction-pairs and junction-loops.

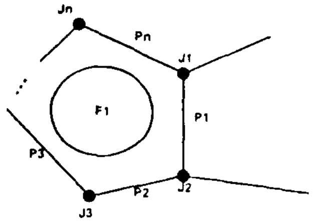


Figure 3: Labeling a face from its boundary vertices and edges.

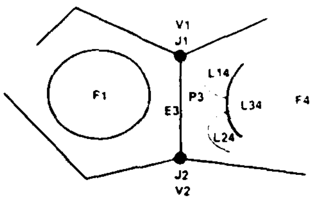


Figure 4: Labeling many faces joined by common vertices and edges.

- *If the vertices and edges of a face boundary have consistent labelings, then the face boundary has at least one consistent labeling.* Starting from a vertex or an edge, let's walk clockwise keeping the face on our right-hand side (Figure 3).

  Let's pick a junction J_1 at vertex V_1 from its set of junctions. Since V_1 is locally consistent with labeling J_1, we can pick a junction-pair P_1 for edge E_1 such that $J_1 = aref(P_1, V_1)$. Next, edge E_1 is locally consistent with labeling P_1, so we can pick junction J_2 at the next vertex V_2 such that $aref(P_1, V_2) = J_2$. By induction, we find a chain of junctions and junction-pairs that represents a labeling for the face boundary.

  If the face boundary is open, we could start with V_1 being the first vertex of the boundary. By induction from one vertex/edge to the next, we automatically get a consistent labeling of the face boundary if Waltz's filtering has been done. If the face boundary is closed, we have to check that the first and last vertices V_1 and V_n can be labeled consistently at both ends of edge E_n. Specifically, we must have $aref(P_n, V_1) = J_1$. This is true if and only if the face has junction-loops. So Waltz's filtering captures labeling constraints over arbitrary open chains, whereas junction-loops capture constraints over both open and closed chains, going around the faces of the image.

- *If a face and its neighboring faces, as well as their boundary vertices and edges, all have consistent labelings, then there exists at least one consistent labeling for all these faces, edges, and vertices.* Let face F_1 be the starting face with junction-loop L_1 found from above. Let's find a face labeling for each neighboring face of F_1, starting from the vertices and edges on the boundary of F_1 (Figure 4).

Let's pick junction J_i of vertex V_i such that $arcf(L_1, V_i) = J_i$. Since vertex V_i is locally consistent, we can pick the junction-loops L_{ij} for the adjacent faces F_{ij} connecting at V_i such that $J_i = arcf(L_{ij}, V_i)$.

So the labelings for the neighboring faces of F_1 exist. However, for some faces such as F_4, we might have different face labelings from the two vertices V_1 and V_2. We have to check that there exist two junction-loops L_{14}, L_{24} having the same labeling at least local to the vertices V_1, V_2, and to the edge E_3. Equivalently, the two junction-loops must have the same pair of junctions (J_1, J_2) for (V_1, V_2). This is true if and only if the junction-pair $P_3 = (J_1, J_2)$ of edge E_3 is locally consistent with a junction-loop L_{34} of face F_4. So a junction-pair is a more complete description of the local labeling at an edge than an edge-label.

We have shown that the labeling for all faces joined with F_1 exists and is locally consistent with the face boundary δF_1. If these adjacent faces share further edges and vertices, we can use the previous induction, going from one vertex/edge to the next, starting from the boundary vertices and edges of δF_1, to show that any two joining faces will have two face-labelings that are locally consistent at all the vertices and edges common to the two faces.

- *If all the nodes in the image have consistent labelings, then there exists at least one global labeling for the whole image.* The segmentation of the image into faces guarantees that labeling all the faces will ultimately label the whole image. Just like putting together a puzzle, we label the whole image by fitting the face-labelings with each other. We use the previous induction to label consistently from one face to its neighboring faces, covering all the faces of the image. The induction uses a common junction or junction-pair if two boundaries share a common vertex or edge. ■

Since CSP rejects only the inconsistent local labelings at a node, any globally consistent labeling must be found from the above inductive enumeration. This means that the network of nodes, with labels and local labelings attached to each node, can enumerate and so "contains" all globally consistent labelings.

Finally, let's compare the above findings with more general results from the consistency o' networks of constraints (Montanari 1976, Mackworth 1977, Freuder 1978), and from topology (Hocking and Young 1961, Henle 1979). Recall that image labeling used to be called line labeling, because the core problem is to label all the lines or edges in the image, satisfying all the constraints at the vertices and edges. Junctions are used to capture the n-ary constraints among edges intersecting at a common vertex. A vertex-node is needed to store these junctions. The n-ary constraints among intersecting edges become binary *constraints* between *these edges and the* common vertex. Similarly, by creating higher level k-nodes and enumerating all the locally consistent labelings for these k-nodes, one can synthesize more extended k-ary constraints, covering finally all the edges in the image (Fre :der 1979). At the k-th step, the k-ary constraints become binary constraints between the k-nodes and the $k - 1$-nodes. So, arc consistency algorithms, such as Waltz's filter, can be used to insure k-satisfiability. The automatic synthesis of the implicit n-ary constraints is exponential, and is equivalent, in the worst case, to depth-first search over the whole image. Montanari (1977) showed that making explicit all the implicit n-ary constraints in an arbitrary network of unary and binary constraints is equivalent to finding all the minimal equivalent networks, and is exponential.

Our finding is not in conflict with the above general results for arbitrary networks. But note that line-drawings and images are not arbitrary networks. From topology (Hocking and Young 1961, Henle 1979), it is well known that by introducing faces, we get not only a planar graph but a complete tesselation of the plane. It is the tesselation of the plane that allows global consistency to be deduced from transitive closure of local consistencies. In other words, the planar topology makes explicit all the implicit local n-ary constraints, and global search is never needed. The local n-ary constraints are constraints between edges that intersect at a vertex, edge, and face. These n-ary constraints are captured in the enumeration of all locally consistent volumes around the vertex, edge, and face.

The above parallel labeling algorithm can be extended to find local/global consistency in a network of nodes embedded in an arbitrary d-dimensional tesselation. Exploiting the tesselation of the 3D volume into blocks, Markowsky and Wesley (1980) fleshed out wire-frames into polyhedra through the enumeration of all topologically consistent vertices, edges, faces, and blocks. 2- and 3-dimensional embedded tesselations are common in VLSI designs, because of the physical constraints inherent in the layout and connections between the VLSI components. Parallel CSP can be used, for example, to simulate, analyze, and debug circuit layouts.

5. Ideal Versus Real Polyhedral Images

Figure 5 show the labeling of two blocks, one on top of the other. The face surrounding the two blocks has 16 junction-loops, corresponding to the blocks floating in air, or resting against some imaginary surface at some of its bounding edges. The two interpretations correspond to the face labeled as image background (B) or as polyhedral face (F). Recognizing from the 16 junction-loops that the surrounding face is a touchable background reduces the 16 junction-loops to 1 and further restricts the labeling of the blocks.

The labeling of the jeep with 122 vertices, 167 edges, and 47 faces, takes about 15 seconds on a Symbolics machine running compiled code, Figure 6. Note the drastic reduction from a search space of approximatively $4^{167} \approx 10^{101}$ possible labelings of the whole image (167 edges and each edge has 4 labels) to a solution space of 96 globally consistent labelings, represented compactly by one single network of nodes with all locally consistent labelings.

The labeling of line drawings and projections of ideal polyhedral scenes is fast and local. It is true that Waltz's filtering has the most drastic reduction of the search space, from 10^{80} to 10^{10} for the jeep example. However, without the enumeration of the junction-pairs and junction-loops, and the sufficiency of local consistency, the labeling of the jeep would run for hours, and output billions of globally consistent labelings, each is different from the next at only one edge-label.

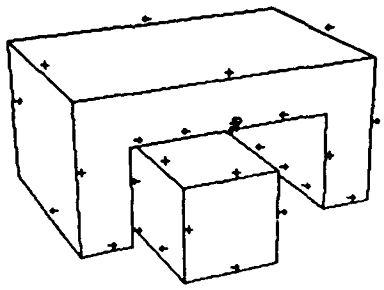


Figure 5: Labeling of the blocks.

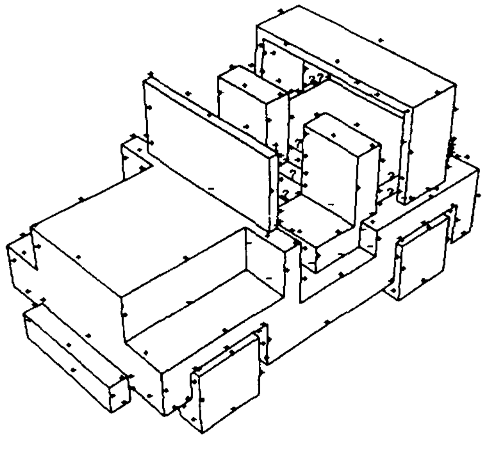


Figure 6: Labeling of the jeep.

Labeling of real polyhedral images proves to be less successful because of unreliable junctions in real segmentations. Figures 7 show a 268x200 image of a V-block, the step edges detected from a Canny edge detector (1986), the segmentation from extending or eliminating the step edges, and the locally consistent labels at all the edges. Note that the step contours are often broken because of thresholding and the fast change in direction of the gradient near corners. To get closed boundaries, we either extend the steps looking for zero-crossings of the second derivative along the gradient direction, or delete dangling step chains. This heuristic is consistent with the facts that zero-crossing contours are always closed, and that there are no dangling edges and vertices in the projection of a polyhedral scene. However, arbitrary extensions and deletions of the steps can lead to tiny loops. By pure coincidence, the tiny loops do not hurt the labeling of the V-block, because they either are disjoint from the V-block, or touch this later at T or ? junctions.

It is to no surprise that such extension and elimination of the steps could create many wrong junctions, missing/extraneous edges and faces, and completely fool the labeling which is based so much on a locally correct segmentation. Figures 8, show a 283x209 image of plastic letters (A, W, E, N), its segmentation and labeling. To avoid as much reflection as possible, we have sand-blasted the plastic blocks to make their surface matte, used a diffuse light source, and reduced from an original image twice as large. We have kept the shadows on purpose, although they are not accounted by the current label dictionary. As expected, most of the damage to labeling is caused not by shadows but by wrong local segmentations and by small fluctuations in the shape of the junctions. However, it is worth to notice that the elimination of global depth-first search highlights the locality of labeling, and opens up the possibility of correcting the local segmentation based on locally consistent labelings. Any corrections should be top-down, and based on significant faces, edges, and vertices, to avoid small errors in the local segmentations.

Acknowledgments

The work reported here is partially funded by General Electric Aerospace Business Group. I would like to thank Joseph Mundy, Deepak Kapur, and David Cyrluk for many discussions, and Tomás Lozano-Peréz for pointing me to papers on network consistency.

References

Canny, J. (1986) "A Computational Approach to Edge Detection." IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. 8. No. 6, November.

Clowes, M.B. (1971) "On Seeing Things," *Artificial Intelligence*. Vol. 2, No. 1, pp. 79-116.

Freuder, E.C. (1978) "Synthesizing Constraint Expressions," *Communications of the ACM*, Vol. 21. No. 11.

Henle, M. (1979) *A Combinatorial Introduction to Topology*. Freeman and Co., San Francisco.

Hocking, J.G., G.S. Young (1961) *Topology*. Addison-Wesley Co., Reading.

Huffman, D.A. (1971) "Impossible Objects as Nonsense Sentences," *Machine Intelligence*, Vol. 6, pp. 295-323.

Mackworth, A.K. (1977) "Consistency in Networks of Relations," *Artificial Intelligence*, Vol. 8, pp. 99-118.

Mackworth, A.K., E.C. Freuder (1985) "The Complexity of some Polynomial Network Consistency Algorithms for Constraint Satisfaction Problems," *Artificial Intelligence*, Vol. 25, pp. 65-74.

Malik, J. (1985) "Interpreting Line Drawings of Curved Objects," *PhD Thesis*, Stanford University.

Markowsky, G., M.A. Wesley (1980) "Fleshing out Wire Frames," *IBM Journal of Research and Development*, Vol. 24, No. 5.

Montanari, V. (1976) "Network of Constraints: Fundamental Properties and Applications to Picture Processing" *Information Science*, Vol. 7, pp. 95-132.

Nguyen, V. (1987) "Exploiting 2D Topology in Labeling Polyhedral Images" Proc. of the 10th Int. Joint Conf. on Artificial Intelligence, Milan, Italy.

Waltz, D.L. (1975) "Understanding Line Drawings of Scenes with Shadows," *The Psychology of Computer Vision*, P.H. Winston (ed), McGraw-Hill, New York, pp. 19-91.

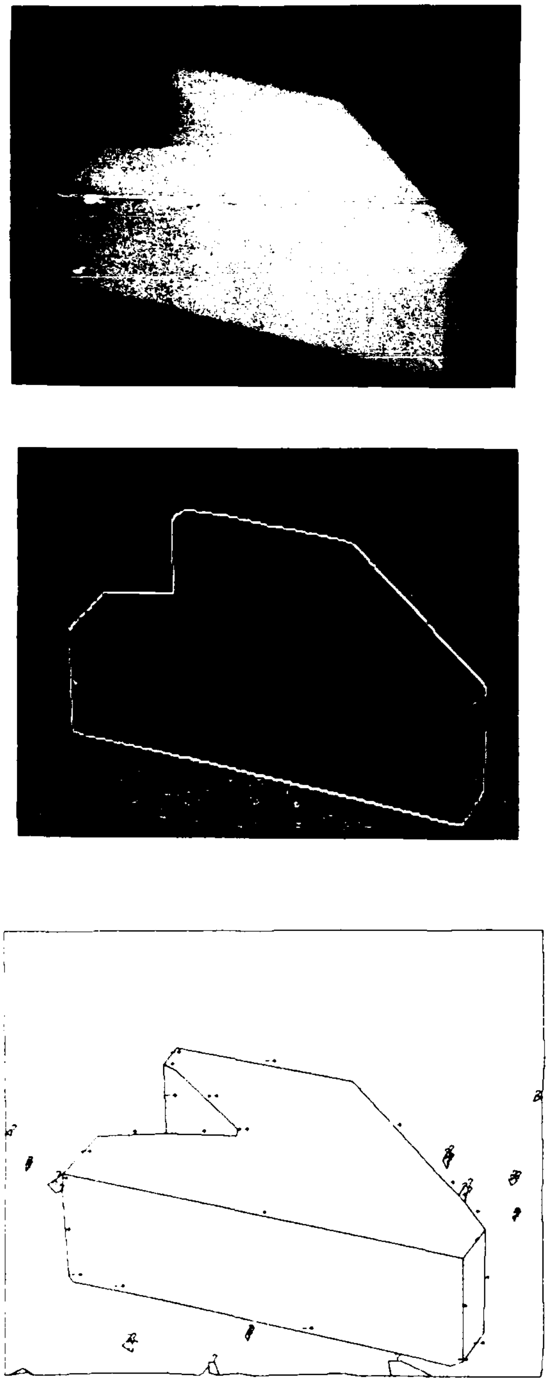


Figure 7: A V block: 1) intensity image, 2) step edges, 3) segmentation and labeling.

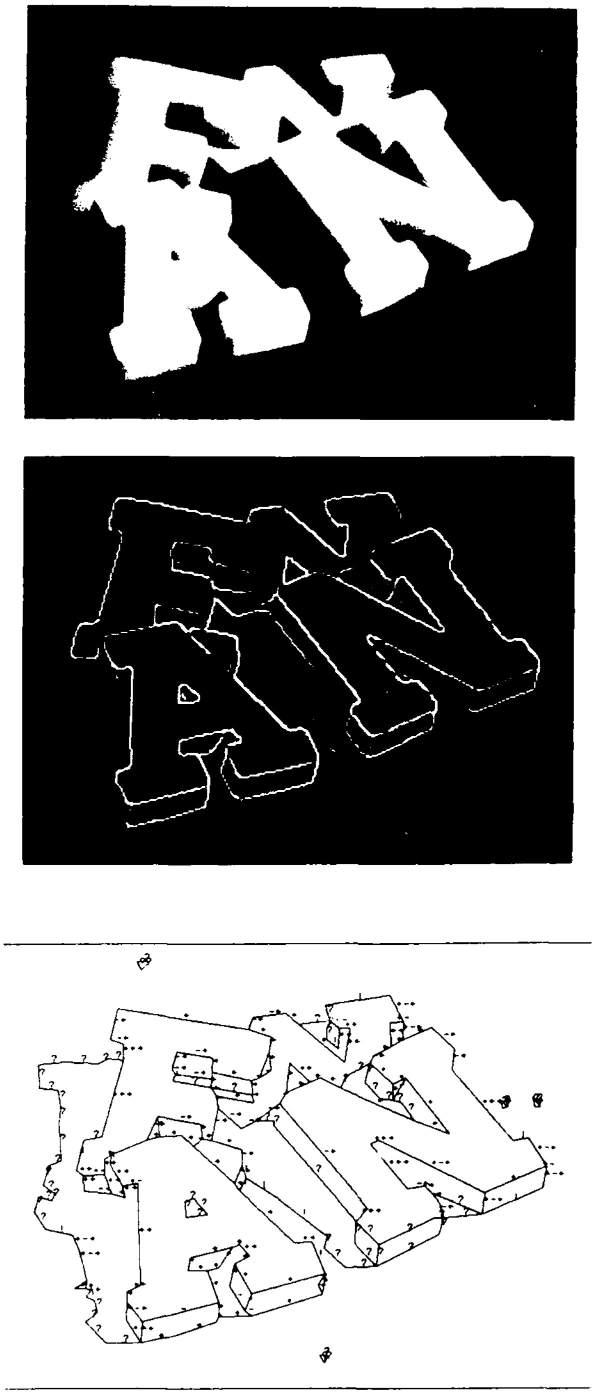


Figure 8: Letters A W E N: 1) intensity image, 2) step edges, 3) segmentation and labeling.